Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems

DISSERTATION

Todd A. Sriver B.S., M.S.
Major, USAF

AFIT/DS/ENS/04-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENS/04-02

Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

In Partial Fulfillment for the Degree of

Doctor of Philosophy

Specialization in: Operations Research

Todd A. Sriver B.S., M.S.
Major, USAF

September, 2004

Sponsored by the Air Force Office of Scientific Research

AFIT/DS/ENS/04-02

Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems

Todd A. Sriver B.S., M.S.

Major, USAF

Approved:

[illegible]
Dr. James W. Chrissis
Committee Chairman

8 Sep 04
Date

[illegible]
Dr. Richard F. Deckro
Committee member

9/8/04
Date

[illegible]
Dr. John O. Miller
Committee member

9/8/04
Date

[illegible]
Lt Col Mark A. Abramson
Committee member

8 Sep 04
Date

[illegible]
Dr. Paul I. King
Dean's Representative

8 Sep 04
Date

[illegible]
Dr. Robert A. Calico
Dean, School of Engineering and Management

AFIT/DS/ENS/04-02

Abstract

A new class of algorithms is introduced and analyzed for bound and linearly constrained optimization problems with stochastic objective functions and a mixture of design variable types. The generalized pattern search (GPS) class of algorithms is extended to a new problem setting in which objective function evaluations require sampling from a model of a stochastic system. The approach combines GPS with ranking and selection (R&S) statistical procedures to select new iterates. The derivative-free algorithms require only black-box simulation responses and are applicable over domains with mixed variables (continuous, discrete numeric, and discrete categorical) to include bound and linear constraints on the continuous variables. A convergence analysis for the general class of algorithms establishes almost sure convergence of an iteration subsequence to stationary points appropriately defined in the mixed-variable domain. Additionally, specific algorithm instances are implemented that provide computational enhancements to the basic algorithm. Implementation alternatives include the use of modern R&S procedures designed to provide efficient sampling strategies and the use of surrogate functions that augment the search by approximating the unknown objective function with nonparametric response surfaces. In a computational evaluation, six variants of the algorithm are tested along with four competing methods on 26 standardized test problems. The numerical results validate the use of advanced implementations as a means to improve algorithm performance.

Acknowledgments

I express my heartfelt gratitude to many individuals who directly or indirectly supported me in the challenging, but rewarding, endeavor of completing a Ph.D. program. I begin with the most important person – my wife – whose strength and encouragement were essential to my success. I am also grateful for our three children (who make me very proud) for the ability to make me smile during those times when the road ahead seemed difficult.

I owe a deep thanks my research advisor, Professor Jim Chrissis. His expertise, guidance, and friendship kept me on the right track while enabling me to enjoy the ride. I am also grateful to the remainder of my research committee, Lt Col Mark Abramson, Professor Dick Deckro, and Professor J. O. Miller for all of the positive support they provided me. In particular, Lt Col Abramson's help on some of the more theoretical issues was crucial to the development of the material presented in Chapter 3. I also thank the Air Force Office of Scientific Research for sponsoring my work.

I would be negligent if I did not acknowledge my friends and colleagues in the B.A.R.F. cubicle farm. The synergy amongst the Ph.D. students in that small building led to some novel ideas incorporated in my research (Major Trevor Laine gets credit for introducing me to kernel regression); but, perhaps more importantly, the moments of levity kept things in the proper perspective.

Finally, I offer a special thanks to both my mother and father. Their love and guidance throughout my life has inspired me to seek achievement.

Todd A. Sriver

Table of Contents

Page

List of Figures

Figure Page

List of Tables

Pattern Search Ranking and Selection Algorithms for Mixed-Variable Optimization of Stochastic Systems

Chapter 1 - Introduction

1.1 Problem Setting

Consider the optimization of a stochastic system in which the objective is to find a set of controllable system parameters that minimize some performance measure of the system. This situation is representative of many real-world optimization problems in which random noise is present in the evaluation of the objective function. In many cases, the system is of sufficient complexity so that the objective function, representing the performance measure of interest, cannot be formulated analytically and must be evaluated via a representative model of the system. In particular, the use of simulation is emphasized as a means of characterizing and analyzing system performance. The term *simulation* is used in a generic sense to indicate a numerical procedure that takes as input a set of controllable system parameters (design variables) and generates as output a *response* for the measure of interest. It is assumed that the variance of this measure can be reduced at the expense of additional computational effort, *e.g.*, repeated sampling from the simulation.

Applications involve the optimization of system designs where the systems under analysis are represented as simulation models, such as those used to model manufacturing systems, production-inventory situations, communication or other infrastructure networks, logistics support systems, or airline operations. In these situations, a search methodology is used to drive the search for the combination of values of the design variables that optimize a system measure of performance. A model of such a stochastic optimization methodology via simulation is depicted in Figure 1.1.

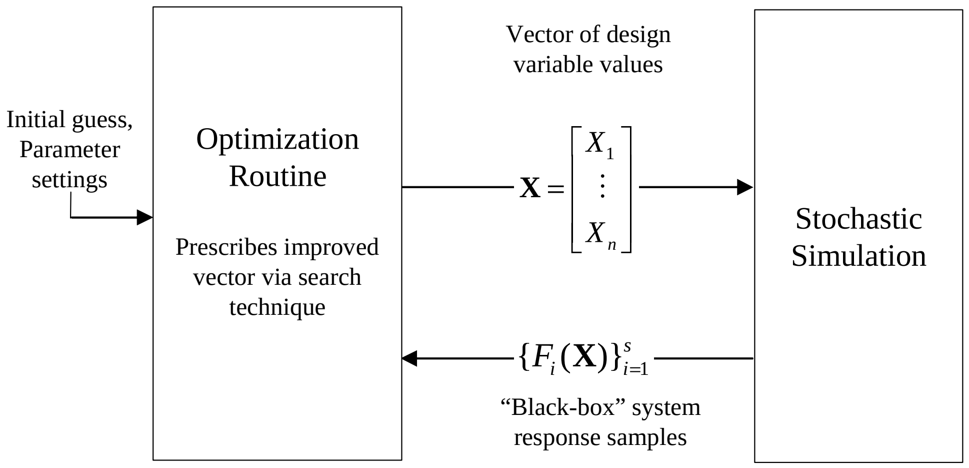

Initial guess, Parameter settings
Optimization Routine
Prescribes improved vector via search technique
Vector of design variable values
$\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$
Stochastic Simulation
$\{F_i(\mathbf{X})\}_{i=1}^{s}$
“Black-box” system response samples

Figure 1.1. **Model for Stochastic Optimization via Simulation**

The random performance measure may be modeled as an unknown *response function* $F(x, \omega)$ which depends upon an n-dimensional vector of controllable design variables $x \in \mathbb{R}^n$, and the vector ω, which represents random effects inherent to the system. The *objective function* f of the optimization problem is the expected performance of the system, given by

$$f(x) = \mathbb{E}_P[F(x, \omega)] = \int_{\Omega} F(x, \omega) P(d\omega), \tag{1.1}$$

where $\omega \in \Omega$ can be considered an element of an underlying probability space $(\Omega, \mathcal{F}, P)$ with sample space Ω, sigma-field $\mathcal{F}$, and probability measure P. It is assumed that the probability distribution that defines the response $F(x, \omega)$ is unknown but can be sampled.

Even for noise-free system responses obtained via simulation, finding optimal solutions using traditional optimization approaches can be difficult since the structure of f is unknown, analytical derivatives are unavailable, and numerical evaluation of f may involve expensive simulation runs. The presence of random variation further complicates matters because f cannot be evaluated exactly and derivative approximating techniques, such as finite differencing, become problematic. Estimating f requires the aggregation of repeated samples of the response F, making it difficult to determine conclusively if one design is better than another and further hindering search methods that explicitly rely on directions of improvement. Multiple samples at each design point implies the necessity of extra computational effort to obtain sufficient accuracy, thereby reducing the number of designs that can be visited given a fixed computational budget.

Additional complications arise when elements of the design vector are allowed to be non-continuous, either discrete-numeric (*e.g.* integer-valued) or *categorical*. Categorical variables are those that can only take on values from a predefined list that have no ordinal relationship to one another. These restrictions are common for realistic stochastic systems. As examples, a stochastic communication network containing a buffer queue at each router may have an integer-valued design variable for the number of routers and a categorical design variable for queue discipline (*e.g.* first-in-first-out (FIFO), last-in-first-out (LIFO) or priority) at each router; an engineering design problem may have a categorical design variable representing material types; a military scenario or homeland security option may have categories of operational risk. The class of optimization problems that includes continuous, discrete-numeric and categorical variables is known as *mixed variable programming* (MVP) problems. In this research, discrete-numeric and categorical variables are grouped into a *discrete* variable class by noting that categorical variables can be mapped to discrete numerical values. For example, integer values are assigned to the queue discipline categorical variable (*e.g.* 1 = FIFO, 2 = LIFO, and 3 = priority) even though the values do not conform to any inherent ordering that the numerical value suggests.

This research considers the optimization of stochastic systems with mixed variables, for which the continuous variable values are restricted by bound and linear constraints. The target problem class is defined as,

$$\min_{x \in \Theta} f(x)\,, \tag{1.2}$$

where $f : \Theta \rightarrow \mathbb{R}$ is a function of unknown analytical form and x is the vector of design variables from the mixed variable domain Θ. This domain is partitioned into continuous and discrete domains Θ^c and Θ^d, respectively, where some or all of the discrete variables may be categorical. Each vector $x \in \Theta$ is denoted as $x = (x^c, x^d)$ where x^c are the continuous variables of dimension n^c and x^d are the discrete variables of dimension n^d. The domain of the continuous variables is restricted by bound and linear constraints $\Theta^c = \{x^c \in \mathbb{R}^{n^c} : l \leq Ax^c \leq u\}$, where $A \in \mathbb{R}^{m^c \times n^c}$, l, $u \in (\mathbb{R} \cup \{\pm\infty\})^{m^c}$, $l < u$, and $m^c \geq n^c$. The domain of the discrete variables $\Theta^d \subseteq \mathbb{Z}^{n^d}$ is represented as a subset of the integers by mapping each discrete variable value to a distinct integer. Furthermore, due to inherent variation in the stochastic system, the function f cannot be evaluated exactly but must be estimated via observations of F obtained from a representative model of the stochastic system. Iterative search methods are necessarily affected by system noise such that the sequence of iterates may be considered random vectors. Hence, the conventional notation X_k to denote a random quantity for the design at iteration k is used to distinguish it from the notation x_k used to denote a realization of X_k.

Relaxation techniques commonly used for mixed-integer problems, such as branch-and-bound, are not applicable to the mixed-variable case because the objective and response functions are defined only at the discrete settings of the categorical variables; therefore, relaxing the "discreteness" of these variables is not possible. Small numbers of categorical variables can sometimes be treated by exhaustively enumerating their possible values, but this approach quickly becomes computationally prohibitive. In this case, a common approach is to conduct a parametric study, in which expert knowledge of the underlying problem is applied to simply select a range of appropriate values, evaluate the objective, and select the best alternative. Of course, this is a sub-optimal approach. Thus, it is desirable to have an optimization method that can treat MVP problems rigorously.

1.2 Purpose of the Research

In designing appropriate solution algorithms for stochastic optimization problems, the following characteristics are considered to be important.

1. **Provably convergent.** Convergent algorithms are desirable to guarantee that a search procedure asymptotically approaches at least a local optimal solution when starting from an arbitrary point. With random error present in objective function evaluation, proving convergence requires additional assumptions and is typically established in terms of probability (*e.g.* almost sure convergence).
2. **General purpose.** To ensure applicability to the widest possible range of problems, the following conditions of an algorithm are desired.
   a. It is valid over all combinations of variable types (*i.e.*, MVP problems).
   b. It requires neither knowledge of the underlying simulation model structure nor modification to its code. Thus, it treats the model as a black-box function evaluator, obtaining an output based on a set of controllable inputs.
3. **Comparatively efficient.** To make the algorithm useful and viable for practitioners, the algorithm should perform well for some subclass of the target problem class (1.2) in comparison to competing methods in terms of some performance measure (*e.g.*, the number of response samples or computer processing time required to achieve a specified improvement in objective function value).

Various sampling-based search strategies have been applied to stochastic optimization problems similar to (1.2). All of the methods discussed in Chapter 2 are deficient with respect to at least one of the aforementioned convergence and/or general-purpose properties. The focus of this research is sharpened by the following statement of the research problem.

1.2.1 Problem Statement

There exists no provably convergent class of methods for solving mixed-variable stochastic optimization problems. Such methods should require neither knowledge of nor modification to the underlying stochastic model. Algorithmic implementations of these methods should account for the practical need of computational efficiency.

1.2.2 Research Objectives

The purpose of this research is to rigorously treat problems of the type (1.2) in their most general form (*i.e.*, mixed variables and black-box simulation) while addressing the need for computationally efficient implementations. The methodology extends a class of derivative-free algorithms, known as *pattern search*, that trace their history to early attempts to optimize systems involving random error. Modern pattern search algorithms are a direct descendent of Box's [33] original proposal to replace regression analysis of experimental data with direct inspection of the data to improve industrial efficiency [144]. Although pattern search methods have enjoyed popularity for deterministic optimization since the 1960's, only within the last decade has a generalized algorithm class been shown to be convergent [143]. Even more recently, the class of *generalized pattern search* (GPS) algorithms has been extended to problems with mixed variables [13] without sacrifice to the convergence theory. Therefore, with respect to convergence and general-purpose properties, GPS algorithms show great promise for application to mixed-variable stochastic optimization problems. To this point in time, these methods have not been formally analyzed for problems with noisy objective functions.

This research extends the class of GPS algorithms to stochastic optimization problems. The approach calls for the use of ranking and selection (R&S) statistical methods (see [140] for a survey of such methods) to control error when choosing new iterates from among candidate designs during a search iteration. Specific objectives of the research are as follows.

1. Extend the GPS algorithmic framework to problems with noisy response functions by employing R&S methods for the selection of new iterates. Prove that algorithms of the extended class produce a subsequence of iterates that converges, in a probabilistic sense, to a point that satisfies necessary conditions for optimality.

2. Implement specific variants within the GPS/R&S algorithm framework that offer comparatively efficient performance. Implementation focuses on two general areas:

    a. The use of modern ranking and selection methods that offer efficient sampling strategies.

    b. The use of surrogate functions to approximate the objective function as a means to accelerate the search.

3. Test the specific methods on a range of appropriate test problems and evaluate computational efficiency with respect to competing methods.

1.3 Overview

This dissertation document is organized as follows. Chapter 2 reviews the relevant literature for sampling-based stochastic optimization and generalized pattern search methods, to include a discussion of convergence and generality properties. Chapter 3 presents a new mixed-variable GPS/R&S algorithmic framework and attendant convergence theory. Chapter 4 describes the specific algorithm options that were implemented for the testing phase of the research. Chapter 5 presents the computational evaluation of algorithm implementations against competing methods over a range of analytical test problems. Chapter 6 offers conclusions, outlines the research contributions, and suggests directions for further research.

Chapter 2 - Literature Review

Prior to investigating new methods to solve mixed-variable stochastic optimization problems, a review of the existing literature is warranted. Section 2.1 surveys methods applied to stochastic optimization problems, with particular emphasis on sampling-based methods because of their applicability to optimization via simulation. The survey focuses on methods that, in some way, address the desired properties outlined in Section 1.2, relative to the target class of problems (1.2). This review demonstrates that a gap exists in the literature for treating general, mixed-variable stochastic optimization problems, which sets the stage for the review of generalized pattern search methods in Section 2.2. Generalized pattern search (GPS) methods, selected as the algorithmic foundation for treating the target problems (1.2) in this research, have been shown to be convergent for mixed-variable deterministic optimization problems in a series of recent results. A chapter summary illustrates the need for rigorous methods to treat mixed-variable stochastic optimization problems and explains why an extension to generalized pattern search is a valid approach for such problems.

2.1 Methods for Stochastic Optimization

Stochastic optimization may be defined in terms of randomness involved in either or both of (a) the evaluation of the objective or constraint functions, or (b) the search procedure itself [134, p. 7]. Throughout this document, stochastic optimization refers to the former. A further distinction is made regarding the methods considered in this section. In particular, methods usually grouped under the heading of *stochastic programming* are not considered. Stochastic programming models typically assume that probability distributions governing the data are known (or can be estimated) [117, p. 7]. This fact is often exploited in constructing effective solution strategies. In the present problem setting, the probability distribution of the response function is assumed to be unknown (although some limited assumptions may be made) but can be sampled.

Most sampling-based methods for stochastic optimization can be grouped into one of five categories: stochastic approximation, random search, ranking and selection, direct search, and response surface methods. Each class of methods is described in the following subsections. A more in-depth account of these and other methods is contained in a number of review articles on *simulation optimization* [9,10,17,34,46,48,63,92,103,119,137,138].

2.1.1 Stochastic Approximation

Stochastic approximation (SA) is a gradient-based method that "concerns recursive estimation of quantities in connection with noise contaminated observations" [83]. In essence, it is the stochastic version of the steepest descent method that rigorously accommodates noisy response functions. These methods possess a rich convergence theory and certain variants can be quite efficient [134, Chap. 7], but apply primarily to continuous domains only, and therefore lack generality.

Early applications of SA to simulation-based optimization appeared in the late 1970s (*e.g.* see [18]) and, since then, has been the most popular and widely used method for optimization of stochastic simulation models [137]. The SA principle first appeared in 1951 in an algorithm introduced by Robbins and Monro [115] for finding the root of an unconstrained one-dimensional noisy function. In general, SA applies to problems with only continuous variables. A multivariate version of the Robbins-Monro algorithm, adapted from [10, p. 317], is shown in Figure 2.1. In the algorithm, the sequence of step sizes a_k (also known as the *gain sequence*) must satisfy restrictions that are critical to the convergence theory.

```
Robbins-Monro Stochastic Approximation Algorithm

Initialization: Choose a feasible starting point $X_0 \in \Theta$. Set step size $a_0 > 0$ and suitable stopping criteria.

Set the iteration counter $k$ to 0.

1. Given $X_k$, generate an estimate $\hat{\gamma}(X_k)$ of the gradient $\nabla f(X_k)$.

2. Compute,
       $X_{k+1} = X_k - a_k \hat{\gamma}(X_k)$ .        (2.1)

3. If the stopping criteria is satisfied, then stop and return $X_{k+1}$ as the estimate of the optimal solution. Otherwise, update $a_{k+1} \in (0, a_k)$ and $k = k + 1$ and return to Step 1.
```

Figure 2.1. **Robbins-Monro Algorithm for Stochastic Optimization (adapted from [10])**

Kiefer and Wolfowitz [68] extended the SA principle to finding the maximum of one-dimensional noisy functions using central finite differences to estimate the derivative. Blum [28] extended the Kiefer-Wolfowitz algorithm to the multi-dimensional case. The use of finite differences to estimate the gradient in Step 1 of the algorithm in Figure 2.1 is often called *finite difference stochastic approximation* (FDSA). Using central differences, the ith element of the gradient is estimated at iteration k according to,

$$\hat{\gamma}_i(X_k) = \frac{\bar{F}(X_k + c_k e_i) - \bar{F}(X_k - c_k e_i)}{2c_k}, \; i = 1, \ldots, n, \tag{2.2}$$

where e_i is the ith coordinate vector and $\bar{F}(X_k \pm c_k e_i)$ denotes an estimate of f at $X_k \pm c_k e_i$ for some perturbation setting $c_k > 0$, perhaps a single sample or the mean of several samples of $F(X_k \pm c_k e_i, \omega)$. Note the reliance of the perturbation parameter c_k on k. As with the gain sequence, the convergence theory relies on restrictions on the sequence c_k.

A disadvantage of finite-differencing is that it can be expensive, requiring response function samples at each of $2n$ design points (using central differences) to estimate the gradient. An alternative, and more efficient, gradient estimator is based on the concept of randomly selecting coordinate directions for use in computing $\hat{\gamma}(x)$. As a generalization of a random direction method proposed in [44], Spall [132] derived the following *simultaneous perturbation* gradient estimator for deterministic response functions,

$$\hat{\gamma}_i(X_k) = \frac{\bar{F}(X_k + c_k d_k) - \bar{F}(X_k - c_k d_k)}{2c_k d_{ki}}, \; i = 1, \ldots, n, \tag{2.3}$$

where $d_k = [d_{k1}, \ldots, d_{kn}]$ represents a vector of random perturbations and $c_k > 0$ has the same meaning as in (2.2). The convergence theory of this approach was subsequently extended to noisy response functions in [133]. Through careful construction of the perturbation vector d_k, the *simultaneous perturbation stochastic approximation* (SPSA) method avoids the large number of samples required in FDSA by sampling the response function at only two design points perturbed along the directions d_k and $-d_k$ from the current iterate, regardless of the dimension n. The perturbation vector d_k must satisfy certain statistical properties defined in [134, p. 183]. Specifically, the $\{d_{ki}\}$ must be independent for all k and i, identically distributed for all i at each k, symmetrically distributed about zero, and uniformly bounded in magnitude for all k and i. The most commonly used distribution for the elements of d_k is a symmetric Bernoulli distribution; *i.e.* ± 1 with probability 0.5 [48].

The efficiency of SA algorithms can be enhanced further by the availability of direct gradients; this led to a flurry of research in more advanced gradient estimation techniques from the mid-1980s through the present day [48]. Specific gradient estimation techniques include Perturbation Analysis (PA) [57], Likelihood Ratios (LR) [52], and Frequency Domain Experimentation (FDE) [124]. These methods often allow an estimate of the gradient with only a single run of the simulation model. However, they require either knowledge of the underlying structure of the stochastic system (for PA and LR) or additional modifications to a model of the system (for FDE) [10]. Therefore, when coupled with SA, they are not considered *sampling-based* methods since the model cannot be treated as a black-box function evaluator.

A well-established convergence theory for sampling-based SA methods dates back to the early work of Kiefer and Wolfowitz [68]. In general, FDSA and SPSA methods generate a sequence of iterates that converges to a local minimizer of f with probability 1 (almost surely) when the following conditions (or similar conditions) are met [47]:

- **Gain sequences**: $\lim\limits_{k\to\infty} a_k = 0$, $\lim\limits_{k\to\infty} c_k = 0$, $\sum\limits_{k=1}^{\infty} a_k = \infty$, and $\sum\limits_{k=1}^{\infty} a_k^2 < \infty$.
- **Objective function regularity conditions**: *e.g.*, continuously differentiable and convex or unimodal in a specified region of the search space.
- **Mean-zero noise**: $\mathbb{E}\left[\hat{\gamma}(X_k) - \nabla f(X_k)\right] = 0$ for all k or in the limit as $k \to \infty$.
- **Finite variance noise**: variance of the noise in $\hat{\gamma}(X_k)$ is uniformly bounded.

The specific mathematical form of these conditions depends on algorithm implementation, assumptions about the problem, and the method of proving convergence. For a coverage of the various approaches to the convergence theory, see [75], [83], or [134, Chap. 4, 6-7]. The restrictions on a_k ensure that the sequence $\{a_k\}$ converges to zero but not so fast as to converge to a sub-optimal value or too slow to avoid any convergence. The harmonic series, $a_k = a/k$ for some scalar a, is a common choice [10, p. 318] for the gain sequence. In practice, the convergence rate is highly dependent on the gain sequence as algorithms may be extremely sensitive to the scalar parameter a such that a few steps in the wrong direction at the beginning may require many iterations to correct [70]. The mean-zero noise requirement ensures that the gradient estimate $\hat{\gamma}_i$ is an unbiased estimate of the true gradient, and the finite variance noise requirement typically ensures that the variance of the noise in the gradient estimate cannot grow any faster than a quadratic function of x [134, p. 106].

Stochastic approximation methods have been modified over the years to enhance performance using step size selection rules to accelerate convergence. One alternative employs a *line search*, a commonly used globalization strategy in deterministic nonlinear programming in which the minimum value of the objective function is sought along the search direction. This has been analyzed for use in SA by Wardi [149], for example, using Armijo step sizes. Another alternative uses *iterate averaging*, which incorporates the use of information from previous iterations and allows the gain sequence $\{a_k\}$ to decrease to zero at a slower rate than $1/k$. The analysis of Polyak and Juditsky [111] and Kushner and Yang [76] shows how the slower decay rate of $\{a_k\}$ can actually accelerate SA algorithm convergence.

Stochastic approximation methods have also been extended to handle more complicated problems. For problems with constraints, the algorithms may be modified by using a *penalty* or a *projection* constraint-handling approach. The penalty approach was analyzed in a FDSA context by Kushner and Clark [75, Sec. 5.1, 5.4] and in a SPSA context by Wang and Spall [148]. Using this approach, the objective function is augmented with the addition of a penalty term,

$$f(x) + r_k P(x)$$

where the scalar $r_k > 0$ increases with k and $P(x)$ is a term that takes on positive values for violated constraints. Penalty terms are well-suited for problems in which some of the constraint functions require noisy response evaluations from the model, since it cannot be determined prior to simulation if a design is feasible with respect to these constraints. However, as in the deterministic case, penalty methods suffer from computational difficulties due to ill-conditioning for values of r_k that are too large [25, p. 369]. Additionally, these methods can produce a sequence of infeasible designs that converge to the optimal (feasible) solution only in the limit, particularly for values of r_k that are too small. If the sampling budget is severely restricted, this can result in a terminal solution with significant constraint violations because the algorithm was not allowed enough of a budget to approach the feasible region.

Projection approaches generate a sequence of feasible design points by replacing (2.1) with

$$X_{k+1} = \Pi_{\Theta}(X_k - a_k \hat{\gamma}(X_k)) \tag{2.4}$$

where Π_{Θ} denotes projection onto the feasible domain Θ. Such methods are analyzed in the FDSA context by Kushner and Clark [75, Sec. 5.3] and in the SPSA context by Sadegh [118]. Projection methods are useful when all constraint functions are defined explicitly in terms of the design variables so that response samples are not wasted in the process of determining feasibility. However, these methods can typically handle only simple constraint sets (*e.g.*, bound and linear constraints) to facilitate mapping a constraint violation to the nearest point in Θ [134, p. 195].

Although primarily applicable to continuous domains, a version of SPSA has been developed for application to discrete domains of only integer-valued variables [50, 51]. The discrete version uses *fixed gains* (*i.e.*, constant a_k and c_k) and approximates the objective function with a smooth continuous function. The fixed step sizes force the iterates to lie on the discrete-valued grid during the entire search.

2.1.2 Random Search

Random search methods sequentially step through the design space in a random manner in search of better solutions. The general algorithm selects a candidate design point probabilistically from the neighborhood of the incumbent design point and chooses the incumbent or candidate as the next iterate based on a specified criteria. An attractive feature of random search methods is that the flexibility of the neighborhood construct allows for the treatment of mixed variables, so they are very general. However, convergent versions of random search exist primarily for discrete-only domains (*e.g.*, [11]).

A general random search algorithm is shown in Figure 2.2. In the algorithm, $\bar{F}(X_k)$ denotes an estimate of $f(X_k)$, perhaps a single sample or the mean of a number of samples of $F(X_k, \omega)$. The algorithm relies on several user-defined features. In Step 1, a candidate is drawn from a user-defined neighborhood $N(X_k)$ of the current iterate X_k. Step 1 also requires the selection of a probability distribution that determines how the candidate is chosen. Appropriate acceptance criteria must be defined in Step 2.

An advantage of random search is that the neighborhood $N(X_k)$ can be defined either locally or globally throughout the design space. In fact, random search is a popular method for global optimization (*e.g.*, see [155]). In either case, $N(X_k)$ must be constructed to ensure the design space is *connected* [11] (*i.e.*, it is possible to move from any point in Θ to any other point in Θ by successively moving between neighboring points). Neighborhood construction depends in large part on the domain Θ. Random search is flexible in that it can accommodate domains that include any combination of continuous, discrete, and

Random Search Algorithm

Initialization: Choose a feasible starting point $X_0 \in \Theta$ and generate an estimate $\bar{F}(X_0)$. Set a suitable stopping criteria.

Set the iteration counter k to 0.

1. Generate a candidate point $X'_k = N(X_k) \in \Theta$ according to some probability distribution and generate an estimate $\bar{F}(X'_k)$.

2. If $\bar{F}(X'_k)$ satisfies acceptance criteria, then set $X_{k+1} = X'_k$. Otherwise, set $X_{k+1} = X_k$.

3. If the stopping criteria is satisfied, then stop and return X_{k+1} as the estimate of the optimal solution. Otherwise, update $k = k + 1$ and return to Step 1.

Figure 2.2. **General Random Search Algorithm (adapted from [11])**

categorical variables. For an entirely continuous Θ, a local neighborhood may be defined as an open ball of a specified radius about the incumbent (*e.g.*, [19, 87]). Alternatively, a global definition may allow a neighbor to assume any value for each design variable within a specified range if the problem's only constraints are variable bounds (*e.g.*, [131]). For an entirely discrete Θ, a local definition for $N(X_k)$ may include the nearest grid points (in a Euclidean sense) from the incumbent (*e.g.*, [7]), whereas a global definition may allow all admissible combinations of discrete settings for the design vector as neighbors (*e.g.*, [8,154]). If Θ has both continuous and discrete components, a hybrid neighborhood structure can be used (see [67] and [120]). Although the random search literature does not appear to explicitly account for categorical variables in a mixed-variable context, the flexibility of neighborhood structures certainly admits such a construct.

Once a neighborhood structure is determined, the method for sampling randomly from the neighborhood must be defined. The simplest approach is a random draw uniformly distributed so that each point in the neighborhood has equal probability of selection [134, p. 38]. This method can be broadly implemented for either continuous or discrete domains. As an alternative example of a local method in a continuous domain, Matyas [87] suggested perturbing the incumbent design randomly, $X_k' = X_k + d_k$, where d_k is distributed normally with a mean zero vector and covariance matrix equal to the identity matrix I_n. That is, each element of the design vector is randomly perturbed from its incumbent value according to a normal distribution with mean zero and unit variance. Such *blind* search methods do not use information learned during the search to improve neighbor selection. Additional methods employ *adaptive* techniques that combine random sampling with knowledge gained during the search to enhance selection.

Matyas [87] suggested a modification to the normally distributed perturbation vector that allows the mean vector and correlation matrix of the perturbations to vary by considering results of preceding iterations. Solis and Wets [131] present a similar method in which the mean of the perturbation vector is a *bias vector* b_k, updated after every iteration, that "slants the sampling in favor of the directions where success has been recorded" [131, p. 25].

The acceptance criteria required in Step 2 of Figure 2.2 are the most critical of the user-defined features in the presence of noisy responses. For the deterministic case, these criteria may simply require improvement in the objective function, $f(X'_k) < f(X_k)$, where $X'_k \in N(X_k)$. Alternatively, moves that fail to yield an improvement may be accepted with a specified probability that decreases with iteration count, such as in *simulated annealing* [49]. Additional considerations are required for noisy response functions to build in robustness to the noise.

Two basic strategies discussed in [134, pp. 50-51] are *averaging* and *acceptance thresholds*. Using averaging, the mean from a number of response samples from the incumbent and the candidate design points are used in place of true function values. The approach more adequately accounts for variation by using an aggregate measure, but adds computational expense. Using thresholding, a candidate design point is accepted if it satisfies $F(X'_k, \omega) < F(X_k, \omega) - \tau_k$, where τ_k is an acceptance threshold. Using a threshold approximately equal to two standard deviations of the estimated response noise implies that only design points with two-sigma improvement are accepted. However, overly conservative thresholds can lead to many rejections and therefore slow convergence.

For continuous domains and noisy response functions, formal convergence proofs for random search methods are rare [134, p. 50]. Yakowitz and Fisher [153, Sect. 4] provide an exception by establishing a convergent method via repeated sampling at design points to minimize the effect of error. For discrete domains with a finite number of points, much recent work has led to several convergent methods. A number of specific methods that include simulated annealing methods are discussed in [11].

In an entirely discrete domain, the random search framework enables the sequence of designs visited to be modeled as a discrete time Markov chain, each iterate representing a state visited by the chain. This fundamental property is key to proving asymptotic convergence as the number of iterations goes to infinity. The strength of the result generally depends on how the optimal solution is estimated; the usual choices being the most frequently visited solution or the current solution under consideration [46].

Methods that estimate the solution using the current design point are able only to show that the sequence of iterates *converges in probability* to an optimal solution; *i.e.*,

$$\lim_{k \to \infty} P\{X_k^* \in \Theta^*\} = 1$$

where $\Theta^* \subseteq \Theta$ is the set of global optimal solutions and $X_k^* \in \Theta$ is the estimate of the optimal solution. In order for this sequence to converge, the methods require statistical evidence that trial moves will result in improvement, where the strength of the evidence grows with the number of iterations [11]. For simulated annealing type algorithms, this is accomplished by decreasing the temperature parameter to zero as iterations increase to infinity. For more traditional random search methods, this is accomplished by forcing candidate solutions to pass an increasing number of trials as iterations accumulate. The number of trials per iteration increases to infinity as iterations increase to infinity.

Methods that use the most frequently visited solution as the estimated optimal solution do not require the progressively conservative moves discussed in the preceding paragraph. In these cases, the sequence of iterates generated by the algorithm do not converge at all (they are irreducible, time-homogeneous, and positive recurrent Markov chains) [11].

However, the sequence defined by $\{X_k^*\}$, where X_ℓ^* is the solution that the Markov chain $\{X_k\}$ has visited most often after ℓ iterations, can be shown to converge *almost surely* to an optimal solution [8]; *i.e.*,

$$P\{\lim_{k\to\infty} I_{\{X_k^*\in\Theta^*\}} = 1\} = 1$$

where the indicator I_A equals one when the event A occurs and zero otherwise. This is a stronger result than convergence in probability.

2.1.3 Ranking and Selection

Ranking and selection (R&S) procedures are "statistical methods specifically developed to select the best system, or a subset of systems that includes the best system, from a collection of competing alternatives" [53, p. 273]. These methods are analogous to exhaustive enumeration of combinatorial optimization problems in which each of a small number (≤ 20) of alternatives can be simulated. Ranking and selection procedures are typically grouped into a larger class of statistical procedures that also includes *multiple comparison* procedures [53]. The coverage of R&S procedures in this literature review results from the fact that they have recently been incorporated within iterative search routines applied to stochastic optimization via simulation, which is also how they are used in this research.

Two general R&S approaches are *indifference zone* and *subset selection* [46]. Indifference-zone procedures guarantee selection within δ of the true best solution with user-specified probability $1 - \alpha$ where δ represents a measure of *practical difference* known as the indifference zone. The parameter δ is called the *indifference zone parameter*. These approaches, using a single stage or multiple stages of sampling, collect response samples from the alternatives, check a certain stopping criteria, then either continue sampling or stop and select the alternative with the smallest response estimate in the final stage [139]. The original procedure by Bechhofer [26] is a single-stage procedure in which the number of samples required of each solution is determined *a priori* according to a tabular value related to the experimenter's choice of δ and α. Bechhofer's method assumed a known and equal variance in response samples across all alternatives. Dudewicz and Dalal [42] and Rinott [114] extended the approach to problems with unknown and unequal response variances by using an initial stage of sampling to estimate variances. These estimates are used to prescribe the number of second-stage samples needed to ensure the probability of correct selection. This concept can be extended to many stages in which the early stages use a predetermined number of samples in order to estimate the number of samples required in the final stage to make a selection. Subset selection is very similar to indifference-zone selection, with the exception that a selected subset of at most m systems will contain at least one system with a response within δ of the optimal value.

To define the requirements for a general indifference-zone R&S procedure, consider a finite set $\{X_1, X_2, \ldots, X_{n_C}\}$ of $n_C \geq 2$ candidate design points. For each $i = 1, 2, \ldots, n_C$, let $f_i = f(X_i) = E[F(X_i, \omega)]$ denote the true objective function value. The f_i values can be ordered from minimum to maximum as,

$$f_{[1]} \leq f_{[2]} \leq \cdots \leq f_{[n_C]}.$$

The notation $X_{[i]}$ indicates the candidate with the ith best (lowest) *true* objective function value. If at least one candidate has a true mean within δ of the true best, *i.e.* $f_{[i]} - f_{[1]} < \delta$ for some $\delta > 0$ and $i \geq 2$, then the procedure is indifferent in choosing $X_{[1]}$ or $X_{[i]}$ as the best. The probability of correct selection (CS) is defined in terms of the δ and the significance level $\alpha \in (0,1)$, as

$$P\{CS\} = P\left\{\text{select } \ X_{[1]} \mid f_{[i]} - f_{[1]} \geq \delta, i = 2, \ldots, n_C\right\} \geq 1 - \alpha, \tag{2.5}$$

where δ and α are user specified. Since $P\{CS\} = \frac{1}{n_C}$ is guaranteed simply by choosing randomly from the alternatives, the significance level must satisfy $0 < \alpha < 1 - \frac{1}{n_C}$.

Traditional multi-stage indifference-zone procedures can be too computationally cumbersome to accommodate a large set of candidates because they are based on the *least favorable configuration* assumption that the best candidate has a true mean exactly δ better than all remaining candidates, which are tied for second best [140]. As a result, the procedures can overprescribe the number of samples required in the final stage in order to guarantee that (2.5) holds. Two recent directions in R&S research reflect attempts to address this issue. The first has been to combine a search strategy with R&S to enable a global search of a possibly large solution space. As examples, Ólafsson [102] and Pichitlamken and Nelson [110] each introduce an iterative technique that combines R&S with a global optimization strategy known as *nested partitioning* (NP), which is used to adaptively search the feasible space of (possibly large) combinatorial problems. In each approach, a discrete time Markov chain analysis is used to show almost sure convergence to a global optimum of the discrete and finite variable space. Ahmed and Alkhamis [4] describe and analyze a globally convergent algorithm that embeds R&S procedures within simulated annealing for optimization over a discrete domain. Boesel *et al.* [29] and Hedlund and Mollaghasemi [56] combine R&S procedures with genetic algorithms.

The second trend has been to invent more modern procedures that, through enhanced efficiency in terms of sampling requirements, can accommodate a larger number of solutions. One such procedure combines subset selection with indifference-zone selection as a means to screen out noncompetitive solutions and then select the best from the survivors. A general theory is presented by Nelson *et al.* [99] that balances computational and statistical efficiency. This approach maintains a probability guarantee for selecting the best solution when using the combined procedure. Another procedure, by Kim and Nelson [69], is a so-called *fully sequential* procedure, which is one that takes one sample at a time from every alternative still in play and eliminates clearly inferior ones as soon as their inferiority is apparent. After an initial stage of sampling, a sequence of screening steps eliminates alternatives whose cumulative sums exceed the best of the rest plus a tolerance level. Between each successive screening step, one additional sample is taken from each survivor and the tolerance level decreases.

Categorical variables are readily handled by modern R&S techniques since all design alternatives are determined *a priori* and corresponding variable values can be set accordingly. However, the limited capacity of R&S restricts the number of solutions that can be considered to a discrete grid of points in the solution space, so that thorough exploration of this space is not possible. The existing provably convergent techniques [4, 102, 110] that combine R&S with adaptive search currently address entirely discrete domains. Continuous variables can be dealt with via a discretization of the variable space, but this can lead to a combinatorial explosion of the search space and an increase in computational expense.

2.1.4 Direct Search

Direct search methods involve the direct comparison of objective function values and do not require the use of explicit or approximate derivatives. For this reason, they are easily adapted to black-box simulation, demonstrating some inherent generality properties. This feature has led to their use as sampling-based methods for stochastic optimization, which is documented in this section. However, direct search methods for stochastic optimization have only considered unconstrained problems with continuous variables. The GPS class of algorithms, which are a subset of direct search methods, are more general than the classical methods covered in this section, but have yet to applied to stochastic problems. Since GPS methods are the cornerstone of this research, they are covered in more detail in Section 2.2.

Interestingly, direct search methods evolved from efforts to optimize systems involving random error. In conjunction with his early work in the field of RSM, Box proposed a method for improving industrial efficiency known as evolutionary operation (EVOP) [33] in the mid-1950s. Intended as a simple tool that could be used by plant personnel, the estimation of regression models was replaced by direct inspection of data according to the "patterns" of the experimental design [144]. Spendley, Hext, and Himsworth [136] suggested an automated procedure of EVOP for use in numerical optimization and replaced factorial designs with simplex designs. Several more direct search methods were proposed in the 1960s [152] and include the well-known direct search method of Hooke and Jeeves [60] and the simplex method of Nelder and Mead [98], which is an extension of the method of Spendley *et al.* At the time, these methods were considered heuristics with no formal convergence theory [1, p. 22]. Research on direct search methods faded during the 1970s and 1980s but were revived in the 1990s with the introduction and convergence analysis of the class of pattern search methods for unconstrained optimization problems by Torczon [143].

In general, traditional direct search methods are applicable to continuous domains and are easily adapted to stochastic optimization because they rely exclusively on response function samples. Perhaps the most frequently used direct search methods for stochastic optimization are the Nelder-Mead simplex search and Hooke-Jeeves pattern search. The Nelder-Mead method conducts a search by continuously dropping the worst point from a simplex of $n+1$ points and adding a new point. A *simplex* is a convex hull of a set of

$n+1$ points not all lying in the same hyperplane in $\mathbb{R}^n$ [24, p. 97]. During a search iteration, the geometry of the simplex is modified by expansion, reflection, contraction, or shrinking operations that are triggered based on the relative rank of the point added at that iteration. Figure 2.3, based on [141, pp. 5-7] depicts the Nelder-Mead search procedure. In the algorithm, $\bar{F}(X_k)$ denotes an estimate of $f(X_k)$, perhaps a single sample or the mean of a number of samples of $F(X_k, \omega)$. The algorithm relies on several parameters that determine how the geometry is refined during the search. Common parameter choices are $\eta = 1$, $\gamma = 2$, $\beta = \frac{1}{2}$, and $\kappa = \frac{1}{2}$.

Nelder-Mead Search

Initialization: Choose a simplex of feasible points $S_0 = \{X_1, X_2, \ldots, X_{n+1}\} \in \Theta$ and generate estimates $\bar{F}(X_1)$, $\bar{F}(X_2)$, $\ldots$, $\bar{F}(X_{n+1})$. Reorder the set S_0 so that $\bar{F}(X_1) \leq \bar{F}(X_2) \leq \cdots \leq \bar{F}(X_{n+1})$. Set reflection parameter η, expansion parameter γ, contraction parameter β, and shrink parameter κ. Set a suitable stopping criteria.

Set the iteration counter k to 0.

1. If necessary, reorder S_k so that $\bar{F}(X_1) \leq \bar{F}(X_2) \leq \cdots \leq \bar{F}(X_{n+1})$. Find the centroid of the n best points in S_k, $X_{\text{cen}} = n^{-1} \sum_{i=1}^{n} X_i$. Generate reflected point $X_{\text{ref}} = (1+\eta)X_{\text{cen}} - \eta X_{n+1}$ and estimate $\bar{F}(X_{\text{ref}})$.

2. If $\bar{F}(X_{\text{ref}}) \geq \bar{F}(X_1)$, then go to Step 3. Otherwise, generate expansion point $X_{\text{exp}} = \gamma X_{\text{ref}} + (1-\gamma)X_{\text{cen}}$ and estimate $\bar{F}(X_{\text{exp}})$. If $\bar{F}(X_{\text{exp}}) < \bar{F}(X_1)$, then set $X_1 = X_{\text{exp}}$; otherwise, set $X_1 = X_{\text{ref}}$. Go to Step 6.

3. If $\bar{F}(X_{\text{ref}}) > \bar{F}(X_n)$, then go to Step 4. Otherwise, set $X_{n+1} = X_{\text{ref}}$ and go to Step 6.

4. If $\bar{F}(X_{\text{ref}}) \leq \bar{F}(X_{n+1})$, then set $X_{n+1} = X_{\text{ref}}$. Otherwise, retain the current X_{n+1}. Go to Step 5.

5. Generate contraction point $X_{\text{con}} = \beta X_{n+1} + (1-\beta)X_{\text{cen}}$ and estimate $\bar{F}(X_{\text{con}})$. If $\bar{F}(X_{\text{con}}) < \bar{F}(X_{n+1})$, then set $X_{n+1} = X_{\text{con}}$; otherwise, shrink the entire simplex toward X_1 by setting $X_i = \kappa X_i + (1-\kappa)X_1$ for $i = 2, \ldots, n+1$. Go to Step 6.

6. If the stopping criteria is satisfied, then stop and return X_1 as the estimate of the optimal solution. Otherwise, set $S_{k+1} = S_k$ and reorder S_{k+1} (if necessary) so that $\bar{F}(X_1) \leq \bar{F}(X_2) \leq \cdots \leq \bar{F}(X_{n+1})$. Update $k = k+1$ and return to Step 1.

Figure 2.3. **Nelder-Mead Search (adapted from [141])**

Although the Nelder-Mead algorithm possesses no general convergence theory [152], it generates a search path with some inherent robustness for problems with noisy responses, due to its reliance on the relative ranks of the vertices in the simplex [142] (as opposed to precise estimates). However, this very feature may also cause the algorithm to terminate prematurely. Premature termination results when large random disturbances in the functional response change the relative ranks of the function values in the simplex and inappropriately affect the scaling steps [23]. An early attempt to mitigate inappropriate scaling was carried out by Barton [20], who compared a variant of the method to three other methods (including an unmodified Hooke-Jeeves search) to minimize functions with random noise using Monte Carlo simulation. In Barton's approach, points are sampled once; however, a reflected point is resampled if it is found worse than the two poorest values from the previous simplex. After resampling, if a different point is the worst, then the new worst point is used in a new reflection operation. Barton and Ivey [22, 23] introduced three Nelder-Mead variants with the goal of avoiding false convergence. The first variant (called S9) simply increases the shrink parameter κ from 0.5 to 0.9. The second variant (RS) resamples the best point after a shrink step before determining the next reflection. The third variant (PC) resamples X_{ref} and X_n if a contraction is indicated in Step 3 in Figure 2.3 and these two points are compared to each other again without reordering the ranks of the remaining points (if they change). If $\bar{F}(X_{\text{ref}}) > \bar{F}(X_n)$ still holds, then contraction is performed as normal; otherwise, X_{ref} is accepted as the new point in the simplex and contraction is bypassed. Based on empirical results, Barton and Ivey concluded that a combination of variants S9 and RS provide statistically significant improvements over the unmodified procedure, reducing the deviation between the terminal solution and known optimal solution relative to the standard method by an average of 15% over 18 test problems.

Tomick *et al.* [142] suggested further modifications to Nelder-Mead for noisy responses. In their approach, each point in the simplex is averaged over m_k samples per iteration where m_k is adjusted from the previous iteration based on a statistical test on the hypothesis of equal response means across all points of the simplex. If the test is accepted (rejected), sample size is increased (decreased) by a constant factor. This method, which includes the shrink parameter increase (S9) of Barton and Ivey [22, 23], reduced the deviation of the terminal solution from the known optimal solution to less than 20% of the starting value for each of the 18 test problems. Finally, Humphrey and Wilson [61, 62] present a Nelder-Mead variant with three phases in which (a) the terminal point from one phase becomes the starting point for the next phase, (b) the distance between initial simplex points decreases geometrically and the shrink parameter increases linearly with each phase, and (c) the solution is taken as the best of the terminal points from the three phases. Each phase represents a restart of the basic Nelder-Mead procedure where the increase in the shrink parameter serves to protect against premature termination. In comparison to the algorithm of Barton and Ivey [22, 23] (that included the RS and S9 modifications) for six test problems with known solutions, this procedure found a more accurate solution for five of them while expending approximately equal computational effort.

The Hooke-Jeeves method conducts a search via a series of *exploratory* and *pattern* moves through the solution space. During a search iteration, exploratory moves are conducted locally along the coordinate axes, and pattern moves are conducted along the direction defined by the starting and ending points of exploratory moves. In a simulation-based application, Nozari and Morris [101] applied the Hooke-Jeeves pattern search in conjunction with the two-stage R&S procedure of Dudewicz and Dalal [42]. In the approach, the R&S procedure is used in the exploratory search step in order to find which candidate along any of the coordinate axes produces the best solution. The direction from the incumbent to the chosen candidate solution is then used as the pattern search direction. Nandkeolyar and Christy [96] implemented a Hooke-Jeeves algorithm with a modified step size update rule in which only statistically significant improvements in the response function are recognized. Pegden and Gately implemented an optimization module using Hooke-Jeeves pattern search into the GASP [106] and SLAM [107] simulation languages. In both implementations, a standard statistical test is used in the comparison of response means before selecting new iterates. The method starts, stops, and continues one long simulation run for each design point, comparing the means of numerous batches, until the difference in means is statistically significant. In addition to Barton [20], Lacksonen [77] evaluated the Hooke-Jeeves method in a comparison with other methods. Lacksonen increased the number of samples for candidate points as the step length parameter decreased in order to improve precision of the estimate but no formal statistical test was used. Sample sizes of one, four, and seven were used for the prespecified step length values, terminating the algorithm when exploratory search failed to find an improving solution.

Direct search methods do not possess a general convergence theory in the stochastic setting, with one notable exception. In [6], Anderson and Ferris introduce a search algorithm that operates on a set of points (called a *structure*) in a continuous variable domain with noisy function evaluations. The algorithm converges almost surely to a stationary point of a uniformly Lipschitz, continuously differentiable objective function. The operations on the structure are similar to those of the Nelder-Mead algorithm but differ in that, for each reflection, expansion, and contraction operation, all points except the best point of the structure are repositioned, whereas, in Nelder-Mead, only a single point is reflected or expanded. A key assumption in algorithm convergence is that the random error in the responses tends to zero faster than the step length (representing the size of the structure). In practice, this is accomplished via increased samples. Interestingly, the authors note that the convergence proof is dependent on the characterization of random error and, in fact, fails in the absence of error. In these cases, they claim that the method is a generalized pattern search method[1] and convergence is guaranteed by the analysis of [143].

2.1.5 Response Surface Methods

Response surface methods (RSM) are broadly defined as "statistical and mathematical techniques useful for developing, improving, and optimizing processes" [94, p. 1]. When used for optimization, these methods fit a smooth surface to response values obtained from a sampling of design points. This surface, called a *response surface*, *metamodel*, or *surrogate function*, may be searched inexpensively using traditional deterministic methods in order to explore the search space. Since constructing response surfaces depends on the availability of response samples, RSM is easily applied to black-box simulation. These methods can be applied directly to solve stochastic optimization problems or can be used to augment more rigorous procedures as a means to improve the search. For example, Booker *et al.* [32] describe the use of response surfaces within a pattern search framework as a means to accelerate the search. Since response surfaces will be used in the implementation of the algorithms developed in this research (Section 4.2), a broad coverage of RSM for stochastic optimization is presented in this section.

Due to the breadth of its application, the research literature on RSM is vast. In application to stochastic optimization via simulation, its history dates back to the early 1970s (*e.g.*, [123,130]). The basic RSM approach calls for solving a sequence of optimization problems in which the true objective function is approximated by a response surface. The

[1]Note: This is true if the incumbent solution is defined as the centroid of the structure and candidate solutions are the surrounding points of the structure.

process begins by establishing a prespecified number of design points in a region of the search space according to an experimental design. Sampled responses are obtained for each point and a local response surface is built in the region. Information from the response surface is used to guide the search to a new region and the process is repeated until reaching a stopping criterion. Alternatively, the response surface can *globally* approximate the true objective function, and intermediate design points sampled during the search can be used to enhance the accuracy of the global approximation. The primary issues involved in the process include [21]:

- the choice of a functional form of the response surface,
- the choice of an experimental design to select points from the design space, and
- the method for assessing of the adequacy of the fitted model (*e.g.*, lack of fit or mean squared error).

To fit a response surface, response samples must be collected from some set of design *sites* (points in the design space) $X_1, \ldots, X_N$. Let $\bar{F}_i$ denote the response at site X_i where $\bar{F}_i$ may represent a single response or the mean of a set of responses. The input/output relationship for the $\{(X_i, \bar{F}_i)\}_{i=1}^{N}$ data points is often modeled as a deviation from the true objective function,

$$\bar{F}_i = f(X_i) + \varepsilon_i, \qquad i = 1, \ldots, N$$

with observation errors ε_i. Methods for fitting the data points to a response can be divided into two general classes, *parametric* and *nonparametric* methods [55, p. 4]. Parametric methods assume the underlying function f has a prespecified functional form (*e.g.*, a polynomial) fully described by a set of parameters. Nonparametric methods make minimal assumptions regarding the structure of f. Examples of each class of methods will be briefly described in the following paragraphs.

Traditional response surface methods (*e.g.*, [94]) typically use parameterized polynomials where regression is used to fit the response surface. These methods typically fit a function $\hat{f}$ using a linear model (2.6) or a quadratic model (2.7),

$$\hat{f}(x) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i, \tag{2.6}$$

$$\hat{f}(x) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} \beta_{ij} x_i x_j \tag{2.7}$$

where the parameters β_0, β_i, β_{ii}, and β_{ij}, $i = 1, \ldots, n$, $j = i, \ldots n$, are determined through least squares regression which minimizes the sum of the squared deviations of the predicted values from the actual values [94]. After the response surface is built, then the method of steepest descent is typically used, where the search direction is chosen as the negative of the gradient. Under the linear model, the gradient is simply $\nabla \hat{f}(x) = [\beta_1, \beta_2, \ldots, \beta_n]^T$, and under the quadratic model, $\nabla \hat{f}(x) = [\frac{\partial \hat{f}}{x_1}, \frac{\partial \hat{f}}{x_2}, \ldots, \frac{\partial \hat{f}}{x_n}]^T$, where $\frac{\partial \hat{f}}{x_i} = \beta_i + 2\beta_{ii} x_i + \sum_{\substack{i=1 \\ i \neq j}}^{n} \beta_{ij} x_j$.

In application to simulation-based optimization, much of the research in polynomial based RSM prior to 1990 is summarized in Jacobson and Schruben [63], in which several improvements are discussed such as screening for variable reduction, allowance for multiple objectives, constraint-handling via the methods of feasible directions and gradient projection, variance reduction via common and antithetic pseudorandom numbers, and the effects of alternative experimental designs. More recently, Joshi, *et al.* [66] introduced gradient deflection and second-order search strategies to the RSM approach. This method retains information from previous iterations and builds knowledge of second-order curvature of the objective function, thereby avoiding the zigzagging experienced by steepest descent. Another approach by Angün *et al.* [12] generalizes the method of steepest descent search direction to multiple responses using an interior point approach with an affine scaling algorithm and projection. The method derives a scale independent search direction and several step sizes that enables the algorithm to reach a neighborhood of the optimum in a few simulation runs. Finally, Abspoel *et al.* [3] present an approach that uses sequential linear programming with move limits [25, p.432] in concert with polynomial regression for problems with random objective functions and constraints over an integer variable domain.

Another parametric model fitting approach is known as *kriging*. The kriging approach builds a response surface via the combination of a fixed function $g(x)$ and departures from the fixed function in the following form [91]:

$$\hat{f}(x) = g(x) + Z(x),$$

where $Z(x)$ is a realization of a stochastic process with mean zero and a spatial correlation function. The underlying model $g(x)$ globally approximates the true function and is typically taken to be a constant but can be a general function with its own parameters. The Gaussian spatial correlation function is given by

$$\mathrm{Cov}\left[Z(X_i), Z(X_j)\right] = \sigma^2 R(X_i, X_j),$$

where X_i and X_j are two of N design sites, σ^2 is the process variance, and R is the $N \times N$ correlation matrix. A commonly used correlation matrix has ones along the diagonals and the following off-diagonal elements [72]:

$$R(X_i, X_j) = \exp\left(-\sum_{k=1}^{n} \theta_k \left|X_i^k - X_j^k\right|^2\right)$$

where θ_k are the parameters used to fit the model and X_i^k is the kth components of sample point X_i. With this function, each predicted point is essentially a linear combination of exponentially decaying functions that are based on the spatial distance between X_i^k and X_j^k.

Kriging models have gained popularity in optimization methods for expensive deterministic simulation (*e.g.*, [32,128,129,147]). Recently they have been applied toward problems involving randomness [65,73].

The use of *artificial neural networks* (ANNs) may also be considered response surface approximation methods. ANNs are modelled after neurons of the human brain and consist of an input layer, an output layer, and a series of hidden inner layers [54]. They can use noisy response samples to approximate arbitrary smooth functions [21]; therefore, they may be considered nonparametric fitting methods. A comprehensive introduction to ANNs can be found in [150].

The presence of inner layers allow the ANN to learn nonlinear relationships between input and output quantities. In an optimization problem, the input quantities represent sampled design point values and the output quantities represent responses. Each neuron in an ANN has an activation function (sigmoid function or step function) with associated weight parameters that are analogous to regression parameters in polynomial regression. The ANN is *trained* on the sampled design points by finding values for the weights that minimize an error function that quantifies the difference between actual response values and values predicted by the ANN. In this manner, the ANN is a predictive tool that produces new output (response) values for new input values. This method has been used, for example, by Laguna and Marti [78], in application to optimization via stochastic simulation of a jobshop. In their approach, the ANN is trained during the course of the search and then used to filter out candidate designs that are predicted to be inferior before expending response samples for those designs.

Another nonparametric fitting method is known as *kernel regression* or *kernel smoothing*. Härdle [55] provides a detailed coverage of these methods, with particular attention paid to the case of noisy responses (see [55, Sect 2.1] for a discussion). The cornerstone of kernel regression is the *Nadaraya-Watson estimator* [95,151], which is used to approximate the objective function at a point x according to,

$$\hat{f}(x) = \frac{\displaystyle\sum_{i=1}^{N} \bar{F}_i K_h(x - X_i)}{\displaystyle\sum_{i=1}^{N} K_h(x - X_i)} \tag{2.8}$$

where K_h is an appropriately selected *kernel function* that depends on parameter h and the distance from x to each design site. As it has its origins in probability density estimation, the kernel function must integrate to unity; *i.e.*, $\int_{-\infty}^{+\infty} K_h(x) = 1$. The estimate $\hat{f}$ can be thought of as the weighted average of all response samples, $\bar{F}_i$, where the weight received by $\bar{F}_i$ depends on K_h, the distance $(x - X_i)$, and the smoothing parameter h. The kernel function K_h determines the "shape" of the weights and h determines the "size" of the weights. For numerical reasons, kernel functions typically take on mound-shaped forms that are zero outside some fixed interval [55, p. 25], such as the parabolic *Epanechnikov kernel* [43] or a Gaussian. Kernel regression has been used iteratively within a stochastic approximation framework (*e.g.*, [97]) for the purpose of recursively estimating the root of a noisy function.

Additional fitting methods have been proposed to approximate functions that will simply be mentioned here. One such method is known as *multivariate adaptive regression splines* (MARS) [64]. This method adaptively selects a set of basis functions for approximating the response function through a forward/backward iterative approach. Another method involves the use of *radial basis functions* [64]. This method uses linear combinations of a radially symmetric function based on the Euclidean distance or a similar metric to approximate the response functions.

There is no general convergence theory for RSM methods. Indeed, this would be difficult to establish in its pure form since optimization is performed on an approximation of the true objective function. Even in the absence of random noise, the approximate model contains inaccuracies which are exacerbated if the true function is highly nonlinear. Furthermore, due to their interpolatory nature, the methods are usually restricted to entirely continuous domains. However, extensions to integer variables are possible by relaxing integrality constraints on the approximate model and ensuring that solutions encountered during the search are mapped to admissible discrete points in the search space (*e.g.*, [3]). This approach is unsuitable with respect to categorical variables, necessitating the construction of an independent response surface for each combination of categorical variable settings, resulting in escalating computational requirements.

2.1.6 Other Methods

A brief mention of other methods used for stochastic optimization is warranted, particularly since most commercially available simulation software packages that offer some optimization functionality do not use the methods of the previous sections. Rather, most packages use *heuristic* search procedures [48, Table 1].

A search heuristic is a "technique which seeks good (*i.e.* near-optimal) solutions at a reasonable computational cost without being able to guarantee feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is [112, p. 6]". Heuristics typically use random or deterministic sampling as a tool to guide exploitive search techniques that are more efficient than pure random sampling [54]. Examples of search heuristics include evolutionary algorithms (genetic algorithms, evolutionary strategies and evolutionary programming), scatter search, tabu search, and simulated annealing. Most heuristics are devised with mechanisms to enable global search and escape local minima; hence, they have found successful application for large, nonconvex, and combinatorial problems. In recent years, the use of search heuristics for stochastic optimization via simulation has grown rapidly, evident by its dominance in software. This, in part, is a reflection of their relative ease of use and generality (they can easily be adapted to mixed-variable problems and require only black-box response samples). However, their application to stochastic problems has been largely unmodified from their original form, relying on inherent robustness to noise rather than explicitly accounting for noise [48].

2.1.7 Summary of Methods

Of the methods presented in this section, stochastic approximation possesses the richest convergence theory. However, since these methods represent a class of gradient-based methods, they are geared toward problems with continuous variables. In theory, problems with a mix of integer and continuous variables (mixed-integer problems) could be addressed via methods that iteratively solve subproblems in which some integrality restrictions are relaxed, such as branch-and-bound. However, there is little evidence from the literature that such approaches have been applied in conjunction with SA and, in fact, only in limited applications has SA been extended to problems with only integer-valued variables [50, 51]. Random search methods are the most general methods that possess some convergence results, but a general convergence theory over a mixed-variable domain has not been established. Ranking and selection procedures inherently provide a sense of convergence via probability guarantees, but if applied unmodified, are only able to accommodate a small, discrete set of designs. Direct search methods that have been applied to stochastic optimization have, thus far, not considered a mixture of variable types nor do they yet possess a general convergence theory. Finally, response surface methods, while useful for modeling and analyzing the input/output relationship between design variables and responses, do not possess convergence properties and apply, in general, to continuous variables only. However, they can provide a useful means to improve more rigorous methods.

This review illustrates the need for convergent algorithms that can treat general, mixed-variable optimization problems with noisy response functions. The generalized pattern search class of algorithms, reviewed in the following section, has been shown to be convergent for deterministic optimization problems in a series of recent results. These methods will provide the basis for a convergent class of algorithms in this research.

2.2 Generalized Pattern Search

In recent years, research in direct search theory has led to several results for the subclass of direct search algorithms known as pattern search. This section describes the various pattern search approaches found in the literature, beginning with the unconstrained case over continuous variables and followed by extensions to more difficult problem settings.

2.2.1 Pattern Search for Continuous Variables

Upon its introduction and convergence analysis, Torczon [143] demonstrated that a generalized class of pattern search methods unifies various distinct pattern search techniques; namely, the Hooke-Jeeves method, coordinate search with fixed step lengths, EVOP with factorial designs [33], and multidirectional search of Dennis and Torczon [38]. Torczon's paper was significant in that it established a global convergence theory without ever computing or explicitly approximating derivatives.

Pattern search algorithms are defined through a finite set of directions used at each iteration. The direction set and a step length parameter are used to construct a conceptual mesh centered about the current iterate (the incumbent). Trial points are selected from this discrete mesh, evaluated, and compared to the incumbent in order to select the next iterate. If an improvement is found among the trial points, the iteration is declared successful and the mesh is retained or coarsened; otherwise, the mesh is refined and a new set of trial points is constructed. Torczon proved that, for a continuously differentiable function f, a subsequence of the iterates $\{x_k\}$ produced by the generalized class of methods converges to a stationary point of f (*i.e.*, $\liminf_{k\to\infty} ||\nabla f(x_k)|| = 0$) by showing that the mesh size (step length) parameter becomes arbitrarily small.

The mesh is defined by a finite set of directions that must be sufficiently rich to ensure that a component of the steepest descent direction can be captured by at least one element of the set when the current iterate is not a stationary point. Lewis and Torczon [79] applied the theory of positive linear dependence [37] to establish criteria for a *core* set of directions. The core direction set must be drawn from a set that *positively spans* the space $\mathbb{R}^n$, where a positive spanning set of directions is defined as one in which nonnegative linear combinations of all directions span $\mathbb{R}^n$. Typically this set forms a *positive basis*, which is the smallest proper subset of a positive spanning set that still positively spans $\mathbb{R}^n$. A positive basis contains between $n + 1$ (a *minimal* set) and $2n$ (a *maximal* set) elements; therefore, the worst case number of trial points per iteration can be bounded to $n + 1$ points by an appropriately constructed direction set.

Lewis and Torczon extend the results of [143] and [79] to problems with bound constraints [80] and linear constraints [81]. In these situations, the set of search directions must be sufficiently rich to ensure that some of the positive spanning directions conform to the geometry of the constraint boundaries. With this construct, when the current iterate is not a constrained stationary point, there is at least one feasible direction of descent from which to choose.

Audet and Dennis [14] present an alternative but equivalent version of pattern search and attendant convergence theory for bound and linear constrained problems. In their analysis, various convergence results are reported that relate the optimality conditions to smoothness properties of the objective function and to the defining directions of the algorithm. It should be noted that Audet and Dennis explicitly separate the search for an improved iterate into a SEARCH and a POLL step. The optional SEARCH step employs a user-defined strategy to seek an improved mesh point. This step contributes nothing to the convergence theory, but allows the user great flexibility to apply any desired heuristic to speed convergence. For example, approaches may include randomly selecting a space-filling set of points using Latin hypercube design or orthogonal arrays, or applying a few iterations of a genetic algorithm. For computationally expensive functions, one common approach is to use previously sampled responses to construct and optimize a less expensive surrogate function on the mesh using the methods of Section 2.1.5. Such methods have been implemented within a pattern search framework without sacrifice to the convergence theory [30–32, 39, 86, 127, 145, 147].

Audet and Dennis [16] extend their approach to nonlinear constraints by implementing a *filter* method [45], which accepts new iterates if either the objective function or an aggregate constraint violation function is reduced. In an alternative approach to nonlinear constraint handling, Lewis and Torczon [82] use an augmented Lagrangian function from Conn, Gould, and Toint [36] to construct a bound constrained subproblem that is solved approximately using a pattern search. Finally, Audet and Dennis [15] recently developed an extension to GPS, known as Mesh Adaptive Direct Search (MADS), that replaces the filter method with a *barrier* method that assigns a value of $+\infty$ to infeasible iterates without evaluating their objective function. The key to MADS is to conduct the POLL step using a dense set of directions that enable the resulting algorithms to retain convergence properties under weak constraint qualifications at the limit point.

2.2.2 Pattern Search for Mixed Variables

A pattern search framework for MVP problems with bound and linear constraints was developed by Audet and Dennis [13] by incorporating user-defined discrete neighborhoods into the definition of the mesh. The methodology was further generalized in [1] and [2] to include nonlinear constraints. In the mixed variable case, the POLL step is conducted by searching a subset of the mesh with respect to the continuous variables and searching a user-defined discrete neighbor set. If the POLL step does not yield an improved solution, an EXTENDED POLL step is initiated in the continuous neighborhood of any discrete neighbor with an objective function value sufficiently close (*i.e.* within a tolerance ξ) to that of the incumbent. This aspect of the algorithm allows extension of the convergence theory to the mixed variable domain but incurs a cost of more function evaluations.

2.2.3 Pattern Search for Random Response Functions

Pattern search applied to stochastic optimization problems is rare. Ouali *et al.* [104] applied multiple repetitions of generalized pattern search directly to a stochastic simulation model to seek minimum cost maintenance policies where costs were estimated by the model. In a more rigorous approach, Trosset [146] analyzed convergence in the unconstrained, continuous case by viewing the iterates as a sequence of binary ordering decisions. By defining $\Lambda_k = f(X_k) - f(Y)$, where Y is a trial point from the mesh, the following hypothesis test,

$$H_0 : \Lambda_k \leq 0 \qquad \text{versus} \qquad H_1 : \Lambda_k > 0 \tag{2.9}$$

accepts Y as the new iterate if the null hypothesis is rejected. Such a test is subject to *Type I* and *Type II* errors. A Type I error is made if H_0 is rejected when it is actually true and occurs with probability α; a Type II error is made if H_0 is accepted when H_1 is true and occurs with probability β. A selection of a sequence of significance levels $\{\alpha_k\}$ such that

$\sum_{k=1}^{\infty} \alpha_k < \infty$ ensures (with probability one) a finite number of Type I errors. In addition, let $\{\lambda_k\}$ be a sequence of alternatives satisfying $\lambda_k > 0$, $\lambda_k = o(\Delta_k)$, and $\lambda_k \to 0$ that require power $1 - \beta_k$ when conducting the test in (2.9). Choosing a sequence $\{\beta_k\}$ such that $\sum_{k=1}^{\infty} \beta_k < \infty$ ensures a finite number of Type II errors when $\Lambda_k \geq \lambda_k$. Hence, Trosset claims that a sequence of iterates from a GPS algorithm can be shown to converge almost surely to a stationary point of f but, in practice, would require a very large number of samples to guarantee convergence [146]. He uses a power analysis, a statistical technique designed to determine the number of samples required to guarantee a probability $1 - \beta$ (known as the *power* of the test) of rejecting H_0 when H_1 is true, to show that the number of samples per iteration grows faster than the squared reciprocal of the mesh size parameter.

2.3 Summary

Section 2.1 reviewed the various approaches to sampling-based stochastic optimization. Each class of methods was discussed with regard to the important properties of convergence and generality. The review illustrates the need for rigorous algorithms that can treat the target class of problems (1.2). The GPS class of algorithms, reviewed in Section 2.2, possesses desirable convergence properties for deterministic problems. Due to its reliance on only response samples (*i.e.*, no derivatives) and as its applicability over mixed variable domains, GPS also possesses desirable generality properties. Extension of pattern search theory to the stochastic setting has only recently been introduced [104, 146], yet has not been thoroughly studied to yield new theoretical or empirical results. In a novel approach that combines pattern search with ranking and selection, the remaining chapters provide results of both a theoretical and empirical nature.

Chapter 3 - Algorithmic Framework and Convergence Theory

This chapter presents the algorithmic framework and convergence theory for mixed variable stochastic optimization with bound and linear constraints on the continuous variables using a combined generalized pattern search with ranking and selection approach. Section 3.1 provides some basic definitions for mixed variable domains. Section 3.2 presents the mathematical framework for construction of the mesh from which candidate solutions are drawn. Section 3.3 addresses the handling of bound and linear constraints within this framework. Section 3.4 summarizes the traditional mixed-variable GPS approach used for deterministic optimization. Section 3.5 discusses the alternative approach to iterate selection in the presence of random responses by selecting from among a number of candidates using R&S. Section 3.6 presents and describes the new class of algorithms, and Section 3.7 provides a theoretical convergence analysis that proves almost sure convergence to an appropriately defined first-order stationary point. Finally, Section 3.8 illustrates a basic version of the algorithm on a simple example.

3.1 Mixed Variables

In order to devise algorithms for the target class of problems, it is important to have notions of local optimality and stationarity for mixed variable domains. Local optimality in a continuous domain is well established, and even for discrete variables, it is not difficult to define. However, since categorical variables typically have no inherent ordering, the concept of a local neighborhood must be defined in the context of the problem. For example, in Kokkolaras *et al.* [74] the optimization problem was to determine the optimal number and types of insulators in a thermal insulation system. Given a design, a discrete neighbor was defined to be any design in which an insulator was replaced with one of a different material, or one in which the number of insulators was increased or decreased by one.

To generalize this for MVP problems, the set of discrete neighbors is defined by a set-valued function $\mathcal{N} : \Theta \to 2^{\Theta}$, where 2^{Θ} denotes the power set of Θ. The notation $y \in \mathcal{N}(x)$ means that the point y is a *discrete neighbor* of x. By convention, $x \in \mathcal{N}(x)$ for each $x \in \Theta$, and it is assumed that $\mathcal{N}(x)$ is finite.

Local optimality in a mixed variable domain can be defined in terms of the set of discrete neighbors. The following definition is due to Audet and Dennis [13].

Definition 3.1 (Local Minimizer) A point $x = (x^c, x^d) \in \Theta$ is a *local minimizer* of f with respect to the set of neighbors $\mathcal{N}(x) \subset \Theta$ if there exists an $\epsilon > 0$ such that $f(x) \leq f(v)$ for all

$$v \in \Theta \cap \bigcup_{y \in \mathcal{N}(x)} (B(y^c, \epsilon) \times y^d). \tag{3.1}$$

This definition is stronger than simply requiring optimality with respect to the continuous variables and also with respect to discrete neighbors. It requires the local minimizer to have lower function value than any point in a neighborhood of each discrete neighbor. Furthermore, the quality of the local minimizer is impacted by the user-defined discrete neighborhood. A larger set of discrete neighbors results in a more global local minimizer, but algorithms that require function evaluations at each discrete neighbor will do so at a greater cost. Since optimization algorithms are rarely guaranteed to converge to local optimizers in general, convergence to a point satisfying certain first-order stationarity conditions is a good substitute. The following definition, which is similar in form to that of [84] for unconstrained problems, is implied but not formally stated in [13] and [1]. The notation $\nabla^c f$ represents the gradient of f with respect to the continuous variables while holding the discrete variables constant.

Definition 3.2 (First-order necessary conditions in mixed-variable domain) A point $x \in \Theta$ satisfies *first-order necessary conditions for optimality* if

1. $(w^c - x^c)^T \nabla^c f(x) \geq 0$ for any feasible $(w^c, x^d) \in \Theta$;
2. $f(x) \leq f(y)$ for any discrete neighbor $y \in \mathcal{N}(x) \subset \Theta$;
3. $(w^c - y^c)^T \nabla^c f(y) \geq 0$ for any discrete neighbor $y \in \mathcal{N}(x)$ satisfying $f(y) = f(x)$ and for any feasible $(w^c, y^d) \in \Theta$.

The converge analysis of Section 3.7 shows that, under reasonable assumptions, certain subsequences generated by the class of algorithms introduced in this chapter converge with probability one (almost surely) to limit points satisfying Conditions 1–3 of Definition 3.2. However, the notions of convergence and continuity in a mixed variable domain are first required. The following two definitions appear in [1], with the first also similar to one in [85].

Definition 3.3 (Convergence, limit point) Let $\Theta \subseteq (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d})$ be a mixed variable domain. A sequence $\{x_i\} \in \Theta$ is said to *converge* to $x \in \Theta$ if, for every $\epsilon > 0$, there exists a positive integer N such that $x_i^d = x^d$ and $\|x_i^c - x^c\| < \epsilon$ for all $i > N$. The point x is said to be the *limit point* of the sequence $\{x_i\}$.

Definition 3.4 (Neighbor Set Continuity) A set-valued function $\mathcal{N} : \Theta \subseteq (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d}) \to 2^{\Theta}$ is *continuous* at $x \in \Theta$ if, for every $\epsilon > 0$, there exists $\varsigma > 0$ such that, whenever $u \in \Theta$ satisfies $u^d = x^d$ and $\|u^c - x^c\| < \varsigma$, the following two conditions hold:

1. If $y \in \mathcal{N}(x)$, there exists $v \in \mathcal{N}(u)$ satisfying $v^d = y^d$ and $\|v^c - y^c\| < \epsilon$.
2. If $v \in \mathcal{N}(u)$, there exists $y \in \mathcal{N}(x)$ satisfying $y^d = v^d$ and $\|y^c - v^c\| < \epsilon$.

Thus, given a convergent subsequence of iterates and a convergent subsequence of its discrete neighbors, continuous $\mathcal{N}$ means that the limit points of the discrete neighbor points are themselves discrete neighbors of the limit point of the iterates [1].

3.2 Positive Spanning Sets and Mesh Construction

Pattern search algorithms produce a sequence of iterates that are selected from a discrete mesh in the search domain. Construction of the mesh relies on the following definitions, due to Davis [37]:

Definition 3.5 (Positive combination) A *positive combination* of the set of vectors $V = \{v_i\}_{i=1}^{r}$ is a linear combination $\sum_{i=1}^{r} c_i v_i$, where $c_i \geq 0$, $i = 1, 2, \ldots, r$.

Definition 3.6 (Positive spanning set, positively span) A finite set of vectors $W = \{w_i\}_{i=1}^{r}$ forms a *positive spanning set* for $\mathbb{R}^n$ if every $v \in \mathbb{R}^n$ can be expressed as a positive combination of vectors in W. The set of vectors W is said to *positively span* $\mathbb{R}^n$.

Definition 3.7 (Positive basis) A positive spanning set of vectors W is said to be a *positive basis* for $\mathbb{R}^n$ if no proper subset of W positively spans $\mathbb{R}^n$.

The motivation for using positive spanning sets in GPS algorithms is encompassed in the following theorem, due to Davis [37].

Theorem 3.8 *(Davis [37]). A set D positively spans $\mathbb{R}^n$ if and only if, for all nonzero $v \in \mathbb{R}^n$, $v^T d > 0$ for some $d \in D$.*

If the gradient vector $\nabla f(x)$ exists at x and is nonzero, then, by choosing $v = -\nabla f(x)$, there exists a $d \in D$ such that $\nabla f(x)^T d < 0$. Thus, at least one element of D is a descent direction which ensures that GPS algorithms can always find improving points when the gradient is nonzero.

In the original paper on pattern search for continuous variables, Torczon [143] defined the mesh as follows:

$$M_k(x_k) = \{x_k + \Delta_k d : d \in \Gamma_k\}, \tag{3.2}$$

where Δ_k is the mesh size parameter at iteration k and $d \in \Gamma_k$ means that d is a column of the direction matrix Γ_k. The matrix Γ_k may be decomposed into two matrices,

$$\Gamma_k = \begin{bmatrix} D_k & L_k \end{bmatrix}, \tag{3.3}$$

where $D_k \in \mathbb{R}^{n \times p}$, $p = 2n$, is a *core* set of directions and $L_k \in \mathbb{R}^{n \times q}$, $q \geq 1$, contains at least the column of zeroes and any additional columns of directions that allow algorithm refinements. Lewis and Torczon [79] redefined the requirements for D_k, restricting p to the range $n+1 \leq p \leq 2n$ and ensuring that D_k forms a positive basis according to Definition 3.7. Before the mesh size parameter is reduced, each mesh point defined by the core set of directions, *i.e.* $\{x_k + \Delta_k d : d \in D_k\}$, must be tried and declared unsuccessful.

Audet and Dennis [13] provide an alternative but equivalent definition for continuous variables using the following mesh construct,

$$M_k(x_k) = \{x_k + \Delta_k D z : z \in \mathbb{Z}_+^{|D|}\}, \tag{3.4}$$

where $\mathbb{Z}_+^{|D|}$ represents a $|D|$-dimensional vector of positive integers. The directions in D form a positive spanning set according to Definition 3.6 and must satisfy the restriction,

$$D = GZ, \tag{3.5}$$

where $G \in \mathbb{R}^{n \times n}$ is a nonsingular generating matrix and $Z \in \mathbb{Z}^{n \times |D|}$. One or more points from (3.4) may be tried for improvement during an optional SEARCH step of their algorithm. If the step does not discover an improved solution, the POLL step is invoked in which points from a *poll set* defined as,

$$P_k(x_k) = \{x_k + \Delta_k d : d \in D_k \subseteq D\}, \tag{3.6}$$

are tested until an improved solution is found or the set is exhausted. Note that D_k is also a positive spanning set and P_k, the set of neighboring mesh points, is a subset of M_k.

For MVP problems the mesh is defined differently, but in a way that reduces to the basic mesh structure of (3.4) if there are no discrete variables. A set of positive spanning directions D^i is constructed for each unique combination $i = 1, 2, \ldots, i_{\max}$, of values that the discrete variables may take, *i.e.*,

$$D^i = G_i Z_i, \tag{3.7}$$

where $G_i \in \mathbb{R}^{n^c \times n^c}$ is a nonsingular generating matrix and $Z_i \in \mathbb{Z}^{n^c \times |D^i|}$. The mild restrictions imposed by (3.5) and (3.7) are necessary for the convergence theory. The mesh is then formed as the direct product of Θ^d with the union of a finite number of meshes in Θ^c, *i.e.*,

$$M_k(x_k) = \Theta^d \times \bigcup_{i=1}^{i_{\max}} \left\{ x_k^c + \Delta_k D^i z \in \Theta^c : z \in \mathbb{Z}_+^{|D^i|} \right\}. \tag{3.8}$$

At iteration k, let $D_k^i \subseteq D^i$ denote the set of poll directions corresponding to the i^{th} set of discrete variable values and define $D_k = \cup_{i=1}^{i_{\max}} D_k^i$. The *poll set* is defined with respect to the continuous variables centered at the incumbent while holding the discrete variables constant. Its form is

$$P_k(x_k) = \left\{ x_k + \Delta_k(d, 0) \in \Theta : d \in D_k^i \right\} \tag{3.9}$$

for some $1 \leq i \leq i_{\max}$, where $(d, 0)$ denotes the partitioning into continuous and discrete variables; 0 means the discrete variables remain unchanged, *i.e.*, $x_k + \Delta_k(d, 0) = (x_k^c + \Delta_k d, x_k^d)$.

3.3 Bound and Linear Constraint Handling

An appropriate means to search regions near the constraint boundaries is necessary to find stationary points that reside there. In this situation, the direction set is required to be sufficiently rich so that the polling directions of the GPS algorithm can be chosen to conform to the geometry of the constraint boundaries. In [80] and [81], Lewis and Torczon show how this can be done for every point in Θ via the inclusion of generators for the *tangent cone* to the feasible region as a subset of directions in the direction set. The concepts of a tangent vector and tangent cone are formalized in the following definition, taken from [100, p. 587].

Definition 3.9 (Tangent, tangent cone) A vector $w \in \mathbb{R}^n$ is *tangent* to Θ at $x \in \Theta$ if, for all vector sequences $\{x_i\}$ with $x_i \to x$ and $x_i \in \Theta$, and all positive scalar sequences $t_i \downarrow 0$, there is a sequence $w_i \to w$ such that $x_i + t_i w_i \in \Theta$ for all i. The *tangent cone* at x is the collection of all tangent vectors to Θ at x.

If the current iterate is within $\varepsilon > 0$ of a constraint boundary, the tangent cone $K^{\circ}(x, \varepsilon)$ may be generated as the polar of the cone $K(x, \varepsilon)$ of outward pointing normals for the constraints within ε of x_k. This is illustrated for two dimensions in Figure 3.1.

Inclusion of the tangent cone generators in the set of directions used by pattern search is sufficient to ensure convergence. An algorithm for computing these directions in the absence of degeneracy is given in [81]. It should be noted that, since the target class of problems is restricted to a finite number of linear constraints, there are only a finite number of tangent cone generators for the entire feasible region, which prevents violation of the finiteness of the direction sets, D^i, $i = 1, 2, \ldots, i_{\max}$. However, this would not hold in the presence of nonlinear constraints, which are not treated in this research.

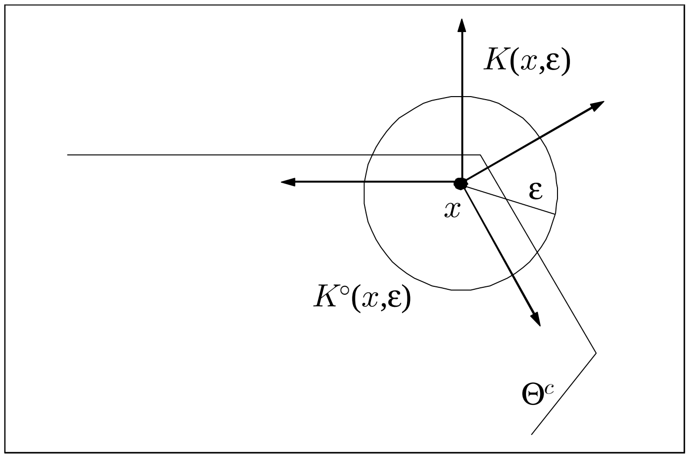


Figure 3.1. **Directions that conform to the boundary of Θ^c (from [81])**

To simplify the convergence analysis in Section 3.7 and avoid reintroducing the method of Lewis and Torczon [81], the following more general definition from [14] is provided. The construction and inclusion of tangent cone generators will be assumed.

Definition 3.10 (Conforming directions) Let D be a positive spanning set in R^n. A rule for selecting the positive spanning sets $D_k = D(k, x_k) \subseteq D$ *conforms to* Θ^c for some $\varepsilon > 0$, if, at each iteration k and for each y in the boundary of Θ^c for which $\|y - x_k\| < \varepsilon$, the tangent cone $K^{\circ}(x, \varepsilon)$ is generated by nonnegative linear combinations of a subset of the columns of D_k.

With conforming directions included, linear constraints can be treated with the simple *barrier* approach. That is, if a linear constraint is violated at a trial point, then a function value of $+\infty$ is assigned without computing the objective function value there, thus saving computational expense.

3.4 The MGPS Algorithm for Deterministic Optimization

Within the GPS framework, mixed variables are accommodated via a user-defined set of discrete neighbors $\mathcal{N}$ introduced in Section 3.1 at each point in the domain. Elements in the neighbor set include the current point and for the remaining elements involve, at a minimum, changes to the values of the discrete variables. For example, if the discrete variables are integers, a neighborhood structure may be defined by holding the continuous variables constant and allowing a maximum change of one unit for only one of the discrete variables, *i.e.*, $\mathcal{N}(x_k) = \{y^c = x_k^c,\, y^d \in \Theta^d : \left\|y^d - x_k^d\right\|_1 \leq 1\}$. This may not be appropriate if the discrete variables are all categorical since the ordering implied by integer values no longer applies; changing a categorical variable value from "1" to "3" may be as valid as a change from "1" to "2". Note that discrete neighbors may require accompanying changes to the continuous variables in order for the solution to make sense for the particular problem.

The basic mixed-variable GPS (MGPS) algorithm for deterministic optimization [13] conducts three distinct searches embodied in the SEARCH, POLL, and EXTENDED POLL steps. At iteration k, the optional SEARCH step evaluates points from a subset of the mesh, $S_k \subset M_k(x_k)$ while the POLL step evaluates points from the set $P_k(x_k)$ and the set of discrete neighbors $\mathcal{N}(x_k)$. Extended polling is conducted after an unsuccessful SEARCH and POLL step for any point $y \in \mathcal{N}(x_k)$ in the discrete neighbor set of the incumbent that satisfies $f(y) < f(x_k) + \xi_k$. The term ξ_k is the *extended poll trigger* at iteration k and must satisfy $\xi_k \geq \xi > 0$ for some positive scalar ξ. The extended poll set of points evaluated about a particular discrete neighbor y_k is denoted as $\mathcal{E}(y_k) = \left\{P_k(y_k^j)\right\}_{j=1}^{J_k}$. Therefore, a poll set with respect to continuous variables is constructed about y_k and the resulting finite number of extended poll points, indexed by a j superscript, are evaluated until an improvement is found over $f(x_k)$ or no further improvement can be made in the continuous variable space near y_k. In either case, let J_k denote the total number of extended poll points considered in the EXTENDED POLL step for discrete neighbor y_k. The point $z_k = y_k^{J_k}$ is termed the *extended poll endpoint*. The set of all extended poll points considered by the EXTENDED POLL step at iteration k is defined as

$$\mathcal{X}_k(\mathcal{E}_k) = \bigcup_{y \in \mathcal{N}_k^\xi} \mathcal{E}(y_k) \tag{3.10}$$

where $\mathcal{N}_k^\xi = \{y \in \mathcal{N}(x_k) : f(x_k) \leq f(y) \leq f(x_k) + \xi_k\}$.

A mixed-variable GPS (MGPS) algorithm for deterministic optimization, due to Audet and Dennis [13], is shown in Figure 3.2. With deterministic function evaluations, the algorithm evaluates trial points from $S_k \cup P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\mathcal{E}_k)$ in search of an improved mesh point. If an improved point is found in any step, the mesh is coarsened or retained; otherwise, if an improved point is not found from the set $P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\mathcal{E}_k)$, the mesh is refined.

The update rules for Δ_k in the algorithm have important implications for the convergence analysis. The mesh is updated (refined, coarsened, or retained) according to the rules found in [1, p. 46]. Refinement must satisfy

$$\Delta_{k+1} = \tau^{m_k^-} \Delta_k \tag{3.11}$$

where $\tau > 1$ is rational and fixed over all iterations, $0 < \tau^{m_k^-} < 1$, and m_k^- is an integer satisfying $m_{\min} \leq m_k^- \leq -1$ for some fixed integer $m_{\min} \leq -1$.

Mixed-Variable Generalized Pattern Search (MGPS) Algorithm

Initialization: Choose a feasible starting point $x_0 \in \Theta$. Set $\Delta_0 > 0$ and $\xi > 0$.

Set the iteration counter k to 0. For $k = 0, 1, 2, \ldots$, perform the following

1. Set extended poll trigger $\xi_k \geq \xi$.

2. SEARCH step (optional): Employ a finite strategy seeking an improved mesh point; *i.e.*, $x_{k+1} \in M_k(x_k)$ such that $f(x_{k+1}) < f(x_k)$.

3. POLL step: If the SEARCH step did not find an improved mesh point, evaluate f at points in $P_k(x_k) \cup \mathcal{N}(x_k)$ until either an improved mesh point x_{k+1} is found or until the set $P_k(x_k) \cup \mathcal{N}(x_k)$ is exhausted.

4. EXTENDED POLL step: If SEARCH and POLL did not find improved mesh point, evaluate f at points in $\mathcal{X}_k(\xi_k)$ until either an improved mesh point x_{k+1} is found or $\mathcal{X}_k(\xi_k)$ is exhausted.

5. Parameter update: If SEARCH, POLL, or EXTENDED POLL finds an improved mesh point, update x_{k+1} and set $\Delta_{k+1} \geq \Delta_k$; otherwise, set $x_{k+1} = x_k$ and $\Delta_{k+1} < \Delta_k$.

Figure 3.2. **MGPS Algorithm for Deterministic Optimization (adapted from [1])**

Coarsening after a successful SEARCH, POLL, or EXTENDED POLL step is accomplished by

$$\Delta_{k+1} = \tau^{m_k^+} \Delta_k \tag{3.12}$$

where $\tau > 1$ is defined as above and m_k^+ is an integer satisfying $0 \leq m_k^+ \leq m_{\max}$ for some fixed integer $m_{\max} \geq 0$.

From these rules, it follows that the mesh size parameter at iteration k may be expressed in terms of the initial mesh size parameter value, *i.e.*,

$$\Delta_k = \tau^{b_k} \Delta_0 \tag{3.13}$$

for some $b_k \in \mathbb{Z}$, which provides for an orderly algebraic structure of the iterates important to proving convergence without imposing a sufficient decrease requirement [143].

3.5 Iterate Selection for Noisy Response Functions

For problems with noisy response functions, single-sample response comparisons of the type used in the algorithm of Figure 3.2 can potentially lead to erroneous decisions due to variation in the response. Alternative techniques for comparing trial points are necessary to ensure that the iterate selection decision accounts for variation and provides some statistical assurances of correct decisions. In the approach of Trosset [146], iterate selection via hypothesis testing is suggested in which a binary selection decision between the incumbent and candidate design is based on sufficient statistical evidence. This approach is generalized in this research by using R&S so that multiple candidates may be considered simultaneously at reasonable computational cost associated with the requisite sampling. This approach provides the following advantages:

- It is amenable to parallelization techniques since several trial solutions can be considered simultaneously in the selection process rather than only two (incumbent and candidate).
- R&S procedures detect the relative order, rather than generate precise estimates, of the candidate solutions. This is generally easier to do [48] and provides computational advantages.
- Selection error is limited to Type II error only, *i.e.*, making an incorrect selection of the best candidate; Type I error is eliminated based on the assumption of a *best* system among the candidates.
- The use of an indifference zone parameter (defined in Section 2.1.3) can be easily and efficiently adapted for algorithm termination.

The mechanics of a general indifference-zone R&S procedures are developed in this section so that this construct may be incorporated into the generalized pattern search algorithm (Section 3.6). At iteration k of the algorithm, consider a finite set $C = \{Y_1, Y_2, \ldots, Y_{n_C}\} \subset M_k$ of candidate solutions, including the incumbent, such that $n_C \geq 2$. For each $q = 1, 2, \ldots, n_C$, let $f_q = f(Y_q) = E[F(Y_q, \cdot)]$ denote the true mean of the response function F. As in Section 2.1.3, the collection of these means can be ordered from minimum to maximum as

$$f_{[1]} \leq f_{[2]} \leq \cdots \leq f_{[n_C]}. \tag{3.14}$$

Again, the notation $Y_{[q]} \in C$ indicates the candidate from C with the qth best (lowest) *true* objective function value and the probability of correct selection is defined as

$$P\{CS\} = P\left\{\text{select } \ Y_{[1]} \mid f_{[q]} - f_{[1]} \geq \delta, q = 2, \ldots, n_C\right\} \geq 1 - \alpha, \tag{3.15}$$

where δ and α become parameters in the algorithm.

Of course, true objective function values are not available in the current problem setting, so it is necessary to work with sample means of the response F. For each $q = 1, 2, \ldots, n_C$, let s_q be the total number of replications and let $\{F_{qs}\}_{s=1}^{s_q}$ be the set of responses obtained via simulation, where $F_{qs} = F(Y_{qs})$, $s = 1, \ldots, s_q$. Then for each $q = 1, 2, \ldots, n_C$, the sample mean $\bar{F}_q$ is computed as

$$\bar{F}_q = \frac{1}{s_q} \sum_{s=1}^{s_q} F_{qs}. \tag{3.16}$$

These sample means may be ordered and indexed the same way as in (3.14). The notation $\hat{Y}_{[q]} \in C$ is used to denote the candidate with the qth best (lowest) *estimated* objective function value as determined by the R&S procedure. The candidate corresponding to the minimum mean response, $\hat{Y}_{[1]} = \arg(\bar{F}_{[1]})$, is selected as the new iterate.

To retain generality of the algorithm class of Section 3.6, Procedure RS(C, α, δ) is defined in Figure 3.3 as a generic R&S procedure that takes as input a candidate set $C \subset M_k$, significance level α, and indifference zone parameter δ, and returns candidate $\hat{Y}_{[1]} = \arg(\bar{F}_{[1]})$ as the best. The technique used in Step 1 to determine the number of samples for each candidate is dependent on the specific procedure. Three specific techniques

Procedure RS(C, α, δ)

Inputs: A set $C = \{Y_1, Y_2, \ldots, Y_{n_C}\}$ of candidate solutions, significance level α, and indifference zone parameter δ.

Step 1: For each candidate Y_q, use an appropriate technique to determine the number of samples s_q required to meet the probability of correct selection guarantee, as a function of α, δ and response variation of Y_q.

Step 2: Obtain sampled responses F_{qs}, $q = 1, \ldots, n_C$ and $s = 1, \ldots, s_q$. Calculate the sample means $\bar{F}_q$ based on the s_q replications according to (3.16). Select the candidate associated with the smallest estimated sample mean, *i.e.*, $\hat{Y}_{[1]} = \arg\left(\bar{F}_{[1]}\right)$ as having the δ-near-best mean.

Return: $\hat{Y}_{[1]}$

Figure 3.3. **A Generic R&S Procedure**

were implemented for the computational evaluation and are described in detail in Section 4.1.

3.6 The MGPS-RS Algorithm for Stochastic Optimization

For stochastic response functions, procedures of the type introduced in Section 3.5 are used within the generalized pattern search framework to select new iterates. This framework is flexible in that a number of specific R&S procedures may be used, so long as they satisfy the probability of correct selection guarantee (3.15).

A mixed variable GPS ranking and selection (MGPS-RS) algorithm is presented in Figure 3.4 for mixed variable stochastic optimization problems with bound and linear constraints on the continuous variables. In the algorithm, binary comparisons of incumbent and trial designs used in traditional GPS methods are replaced by R&S procedures in which one candidate is selected from a finite set of candidates considered simultaneously. The R&S procedures provide error control by ensuring sufficient sampling of the candidates so that the best or δ-near-best is chosen with probability $1 - \alpha$ or greater.

The mesh construct of (3.8) defines the set of points in the search domain Θ from which the candidates are drawn. In the SEARCH step, the flexibility of GPS allows any

Mixed Variable Generalized Pattern Search - Ranking & Selection (MGPS-RS) Algorithm

Initialization: Set the iteration counter k to 0. Set the R&S counter r to 0. Choose a feasible starting point, $X_0 \in \Theta$. Set $\Delta_0 > 0$, $\xi > 0$, $\alpha_0 \in (0, 1)$, and $\delta_0 > 0$.

1. SEARCH step (optional): Employ a finite strategy to select a subset of candidate solutions, $S_k \subset M_k(X_k)$ defined in (3.8) for evaluation. Use Procedure RS($S_k \cup \{X_k\}$, α_r, δ_r) to return the estimated best solution $\hat{Y}_{[1]} \in S_k \cup \{X_k\}$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$ according to (3.12), and $k = k + 1$ and repeat Step 1. Otherwise, proceed to Step 2.
2. POLL step: Set extended poll trigger $\xi_k \geq \xi$. Use Procedure RS($P_k(X_k) \cup \mathcal{N}(X_k)$, α_r, δ_r) where $P_k(X_k)$ is defined in (3.9) to return the estimated best solution $\hat{Y}_{[1]} \in P_k(X_k) \cup \mathcal{N}(X_k)$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$ according to (3.12), and $k = k + 1$ and return to Step 1. Otherwise, proceed to Step 3.
3. EXTENDED POLL step: For each discrete neighbor $Y \in \mathcal{N}(X_k)$ that satisfies the extended poll trigger condition $\bar{F}(Y) < \bar{F}(X_k) + \xi_k$, set $j = 1$ and $Y_k^j = Y$ and do the following.
   a. Use Procedure RS($P_k(Y_k^j)$, α_r, δ_r) to return the estimated best solution $\hat{Y}_{[1]} \in P_k(Y_k^j)$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} \neq Y_k^j$, set $Y_k^{j+1} = \hat{Y}_{[1]}$ and $j = j + 1$ and repeat Step 3a. Otherwise, set $Z_k = Y_k^j$ and proceed to Step 3b.
   b. Use Procedure RS($X_k \cup Z_k$) to return the estimated best solution $\hat{Y}_{[1]} = X_k$ or $\hat{Y}_{[1]} = Z_k$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} = Z_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$ according to (3.12), and $k = k + 1$ and return to Step 1. Otherwise, repeat Step 3 for another discrete neighbor that satisfies the extended poll trigger condition. If no such discrete neighbors remain, set $X_{k+1} = X_k$, $\Delta_{k+1} < \Delta_k$ according to (3.11), and $k = k + 1$ and return to Step 1.

Figure 3.4. **MGPS-RS Algorithm for Stochastic Optimization**

user-defined procedure to be used in determining which candidates from (3.8) to consider. In the POLL step, the entire poll set about the incumbent (3.9) and the discrete neighbor set are considered simultaneously. If SEARCH and POLL are unsuccessful, the EXTENDED POLL step conducts a polling sequence that searches the continuous neighborhood of any discrete neighbor with a response mean sufficiently close to the response mean of the incumbent.

This step is divided into sub-steps to account for the sequence of R&S procedures that may be necessary. In Step 3a, each sub-iterate Y_k^j, indexed by sub-iteration counter j and iteration k, is selected as the best candidate from the poll set centered about the previous sub-iterate using the R&S procedure, terminating when the procedure fails to produce a sub-iterate different from its predecessor. The terminal point of the resulting sequence $\{Y_k^j\}_{j=1}^{J_k}$, denoted as $Z_k = Y_k^{J_k}$ and termed an *extended poll endpoint*, is compared to the incumbent via a separate R&S procedure in Step 3b.

If the *extended poll trigger* ξ_k is set too high, more extended poll steps result, thus making a solution more "global". However, the additional sampling required at the extra points increases computational expense, particularly with high noise levels in the response output.

The algorithm maintains a separate counter for R&S parameters α_r and δ_r to provide strict enforcement of the rules on these parameters that are updated after each execution of the R&S procedure. The rules ensure that each parameter tends to zero as the number of iterations approaches infinity. An additional restriction on α_r is that the infinite series $\sum_{r=1}^{\infty} \alpha_r$ converges; that is, $\sum_{r=1}^{\infty} \alpha_r < \infty$. These restrictions are critical for convergence and are justified in Section 3.7.

The update rules for Δ_k in the algorithm are the same as the deterministic case. Refinement (3.11) is accomplished after SEARCH (if used), POLL, and EXTENDED POLL are all unsuccessful. Coarsening (3.12) is accomplished after any successful SEARCH, POLL, or EXTENDED POLL step.

Each execution of the R&S procedure generates an iterate or sub-iterate that is the candidate returned as the best by the procedure. When the new iterate (sub-iterate) is different from (presumed better than) the incumbent, the iteration (sub-iteration) is termed *successful*; if it remains the same, it is *unsuccessful*. The use of these terms is in keeping with traditional pattern search methods where, in a deterministic setting, a success indicates a strict improvement in the objective function value. Let V_{r+1} denote an iterate or sub-iterate selected from candidate set C of cardinality n_C by the rth R&S procedure of the MGPS-RS algorithm. Each successful and unsuccessful outcome (iteration or sub-iteration) can then be further divided into three cases. These cases follow:

1. The outcome is considered *successful* if one of the following holds:

   a. indifference zone condition is met and R&S correctly selects a new incumbent, *i.e.*,
   $$V_r \neq V_{r+1} = Y_{[1]},\ f(Y_{[q]}) - f(Y_{[1]}) \geq \delta_r,\ q = 2, 3, \ldots, n_C\ ; \tag{3.17}$$

   b. indifference zone condition is met but R&S incorrectly selects a new incumbent, *i.e.*,
   $$V_r \neq V_{r+1} \neq Y_{[1]},\ f(Y_{[q]}) - f(Y_{[1]}) \geq \delta_r,\ q = 2, 3, \ldots, n_C\ ; \tag{3.18}$$

   c. indifference zone condition is not met and R&S selects a new incumbent, *i.e.*,
   $$V_r \neq V_{r+1},\ \left|f(Y_{[q]}) - f(Y_{[1]})\right| < \delta_r \text{ for some } q \in \{2, 3, \ldots, n_C\}\ . \tag{3.19}$$

2. The outcome is *unsuccessful* if one of the following holds:

   a. indifference zone condition is met and R&S correctly selects the incumbent, *i.e.*,
   $$V_r = V_{r+1} = Y_{[1]},\ f(Y_{[q]}) - f(Y_{[1]}) \geq \delta_r,\ q = 2, 3, \ldots, n_C\ ; \tag{3.20}$$

   b. indifference zone condition is met but R&S incorrectly selects the incumbent, *i.e.*,
   $$V_r = V_{r+1} \neq Y_{[1]},\ f(Y_{[q]}) - f(Y_{[1]}) \geq \delta_r,\ q = 2, 3, \ldots, n_C\ ; \tag{3.21}$$

   c. indifference zone condition not met and R&S selects the incumbent, *i.e.*,
   $$V_{r+1} = V_r,\ \left|f(Y_{[q]}) - f(Y_{[1]})\right| < \delta_r \text{ for some } q \in \{2, 3, \ldots, n_C\}\ . \tag{3.22}$$

In the algorithm, X_k and Y_k^j play the role of V_r for iterates and sub-iterates, respectively. Of the possible outcomes for new iterates or sub-iterates, conditions (3.17) and (3.20) conform to the traditional GPS methods for deterministic optimization where, in the case of a successful iteration, a trial point on the mesh has a better true objective function value than the incumbent and, in the case of an unsuccessful iteration, the incumbent has the best true objective function value of all candidates considered. Of particular concern for the convergence analysis are the remaining conditions.

Conditions (3.19) and (3.22) occur when the difference between true objective function values of a trial point on the mesh and the incumbent is smaller than the indifference zone parameter. This situation can result from either an overly relaxed indifference zone or a flat surface of the true objective function in the region of the search. When this occurs, the probability for correct selection cannot be guaranteed. However, forcing convergence of δ_r to zero via update rules ensures that the indifference zone condition will be met in the limit. Of greater concern is the case when the indifference zone condition is met, but the algorithm selects the wrong candidate (*i.e.*, it doesn't choose the candidate with the best true objective function value). This represents conditions (3.18) and (3.21), and occurs with probability α_r or less for the rth R&S procedure. The convergence analysis of the following section addresses controls placed on the errors presented by these conditions.

3.7 Convergence Analysis

In this section, a convergence analysis for the MGPS-RS algorithm is given. The following assumptions are required for the analysis:

A1: All iterates X_k produced by the MGPS-RS algorithm lie in a compact set.

A2: The objective function f is continuously differentiable with respect to the continuous variables when the discrete variables are fixed.

A3: For each set of discrete variables X^d, the corresponding set of directions $D^i = G_i Z_i$, as defined in (3.7), includes tangent cone generators for every point in Θ^c.

A4: The rule for selecting directions D_k^i conforms to Θ^c for some $\varepsilon > 0$ (see Definition 3.10).

A5: For each $q = 1, 2, \ldots, n_C$, the responses $\{F_{qs}\}_{s=1}^{s_q}$ are independent, identically and normally distributed random variables with mean $f(X_q)$ and unknown variance $\sigma_q^2 < \infty$, where $\sigma_\ell^2 \neq \sigma_q^2$ whenever $\ell \neq q$.

A6: The sequence of significance levels $\{\alpha_r\}$ satisfies $\sum_{r=0}^{\infty} \alpha_r < \infty$, and the sequence of indifference zone parameters $\{\delta_r\}$ satisfies $\lim_{r\to\infty} \delta_r = 0$.

A7: For the rth R&S procedure considering candidate set $C = \{Y_1, Y_2, \ldots, Y_{n_C}\}$, Procedure RS($C$, α_r, δ_r) guarantees correctly selecting the best candidate $Y_{[1]} \in C$ with probability of at least $1 - \alpha_r$ whenever $f(Y_{[q]}) - f(Y_{[1]}) \geq \delta_r$ for any $q \in \{2, 3, \ldots, n_C\}$.

A8: For all but a finite number of MGPS-RS iterations and sub-iterations, the best solution $Y_{[1]} \in C$ is unique; *i.e.*, $f(Y_{[1]}) \neq f(Y_{[q]})$ for all $q \in \{2, 3, \ldots, n_C\}$ where $C = \{Y_1, Y_2, \ldots, Y_{n_C}\} \subset M(X_k)$ at iteration k.

These assumptions warrant a brief discussion. Assumption **A1** is a fairly standard assumption, and is easily enforced by including finite upper and lower bounds on the continuous variables, which is very common in practice. Assumption **A3** ensures that the restriction on the direction set (3.7) is maintained in the presence of linear constraints, and assumption **A4** provides for adequate rules to generate conforming directions. A sufficient condition for assumption **A3** to hold is that $G_i = I$ for each $i \in \{1, \ldots, i_{\max}\}$ and the coefficient matrix A is rational [1, p. 73]. The independent, normally distributed requirement for responses from a single alternative in assumption **A5** is common for R&S techniques and is readily achieved in simulation via batched output data or sample averages of independent replications [99]. Furthermore, unequal variances between different alternatives is realistic for practical problems and is readily handled with modern R&S procedures. Assumption **A6** is a requirement levied to enable the convergence proofs in this section. Assumption **A7** provides the correct selection guarantee of the R&S procedure and is required in the absence of identifying a specific method. Most R&S procedures are accompanied by proofs that the correct selection guarantee is met. MGPS-RS is flexible in that any R&S procedure may be used, so long as it satisfies assumption **A7**. Finally, assumption **A8** is required to ensure that the indifference zone condition is eventually met during the course of the iteration sequence. This assumption may seem restrictive, but the likelihood of two candidate mesh points having exactly the same objective function value is quite rare for non-academic problems.

Since MGPS-RS iterates are random variables, the convergence analysis must be carried out in probabilistic terms. To that end, the following definition provides what is needed for iterates in a mixed variable domain, and is consistent with Definition 3.3.

Definition 3.11 (Almost Sure Convergence, Limit Point) Let $\Theta \subseteq (\mathbb{R}^{n^c} \times \mathbb{Z}^{n^d})$ be a mixed variable domain. A sequence of multivariate random vectors $\{X_k\}$ *converges almost surely* (*a.s.*) to the *limit point* $\hat{x} \in \Theta$ if, for every $\varepsilon > 0$, there exists a positive integer N such that $P(X_k^d = x^d) = 1$ and $P\left(\|X_k^c - x^c\| < \varepsilon\right) = 1$ for all $k > N$.

3.7.1 Controlling Incorrect Selections

Random variation in the responses leads to errors in the iterate selection decision in the form of *incorrect selections*. The concept of an incorrectly selected MGPS-RS iterate or sub-iterate was formalized by conditions (3.18) and (3.21). Let $\mathcal{A}_r$ denote the *incorrect selection* event that "V_{r+1} is incorrectly selected by the rth R&S procedure in the MGPS-RS algorithm". For convergence of the iteration sequence $\{X_k\}$, a means of bounding the sequence of incorrect selection events $\{\mathcal{A}_r\}$ is necessary so that the sequence of iterates is not dominated by incorrectly selected (and possibly unimproving) candidates. The restriction on the sequence of significance levels in assumption **A6**, along with the first half of the Borel-Cantelli lemma, provide this means.

Lemma 3.12 *(Borel-Cantelli) Let $\{\mathcal{B}_r\}$ be an infinite sequence of random events. If*

$$\sum_{i=1}^{\infty} P(\mathcal{B}_r) < \infty,$$

then

$$P(\mathcal{B}_r \text{ } i.o.) = 0.$$

The term "*i.o.*" stands for *infinitely often*, so that the event $[\mathcal{B}_r \text{ } i.o.]$ can be interpreted as the event "$\mathcal{B}_r$ happens for infinitely many values of r". Note that there is no requirement for the events $\mathcal{B}_r$ to be independent or identically distributed. A proof of this lemma can be found in [113, p. 102].

Lemma 3.13 *With probability 1, the subsequence of incorrectly selected iterates and sub-iterates generated by algorithm MGPS-RS is finite.*

Proof. Let $\mathcal{A}_r$ denote the occurrence of the event that the rth R&S procedure incorrectly selects the next iterate or sub-iterate. The complement of assumption **A7** yields $P(\mathcal{A}_r) \leq \alpha_r$, $r = 1, 2, \ldots$, and assumption **A6** ensures that $\sum\limits_{r=1}^{\infty} P(\mathcal{A}_r) \leq \sum\limits_{r=1}^{\infty} \alpha_r < \infty$. The result follows directly from Lemma 3.12. ■

The restriction on α_r can be enforced in practice through an appropriately selected update rule. For example, the update rule $\alpha_r = \alpha_0 \rho^r$ for $0 < \rho < 1$ and $\alpha_0 > 0$ results in a geometric series that converges, since $\sum_{r=1}^{\infty} \alpha_r = \frac{\alpha_0}{1-\rho} < \infty$. Using this rule with $\alpha_0 < 1$, as required by the R&S procedure, the rate at which α_r converges to zero can be controlled by the parameter ρ. Values for ρ closer to zero result in faster convergence than those that are closer to one.

A final consideration involving incorrect selections is required to enable analysis of the mesh size parameter. In particular, it is necessary to establish that MGPS-RS cannot cycle indefinitely among iterates that belong to Θ. Such a condition occurs if and only if it is possible to have infinitely many consecutive successful iterations. The following lemma establishes the result.

Lemma 3.14 *With probability 1, the number of consecutive successful MGPS-RS iterations must be finite.*

Proof. Let K_S represent the number of successful iterations of MGPS-RS after iteration k. From conditions (3.17)–(3.19), $K_S = K_\delta + K_C + K_I$ where K_δ is the number of successful iterates until the indifference zone condition is satisfied (3.17), K_S is the number of correctly selected successful iterates (3.18), and K_I is the number of incorrectly selected successful iterates (3.19). Assumptions **A6** and **A8** ensure that $K_\delta < \infty$. Furthermore, since assumption **A1** ensures that all iterates lie in a compact set, it must follow that $K_C < \infty$. Finally, since the number of incorrectly selected successful iterates is a subset of all incorrect selections (successful and unsuccessful), Lemma 3.13 ensures that $P(K_I < \infty) = 1$. It follows that

$$P(K_S < \infty) = P(K_\delta + K_C + K_I < \infty) = P(K_I < \infty) = 1 \ . \qquad \blacksquare$$

3.7.2 Mesh Size Behavior

The main result of this section is that, with probability one, there exists a subsequence of mesh size parameters that goes to zero, *i.e.* $P(\liminf_{k \to +\infty} \Delta_k = 0) = 1$, which is independent of any smoothness assumptions on the objective function. This result was first established by Torczon [143] and subsequently modified for MVP problems by Audet and Dennis [13]. Audet and Dennis later adapted a lemma of Torczon [143] to provide a lower bound on the distance between any two mesh points at each iteration for continuous-variable problems [14], which was then extended by Abramson [1] to MVP problems. This lower bound is stated in Lemma 3.15, the result of which is necessary to show that the mesh size parameter is bounded above in Lemma 3.16. Finally, Theorem 3.17 presents the key result for this section. The proof of Lemma 3.15 is independent of response noise but is included, as found in [1], for completeness. The proofs for Lemma 3.16 and Theorem 3.17 are modified from [1] to account for stochastic responses.

Lemma 3.15 *For any $k \geq 0$, $k \in \mathbb{Z}$, let u and v be any pair of distinct mesh points such that $u^d = v^d$. Then for any norm for which all nonzero integer vectors have norm at least 1,*

$$\|u^c - v^c\| \geq \frac{\Delta_k}{\|G_i^{-1}\|}$$

where the index i corresponds to the combination of discrete variable values defined by $u^d = v^d$.

Proof. From (3.8), $u, v \in M_k(X_k) = \Theta^d \times \bigcup_{i=1}^{i_{\max}} \left\{X_k^c + \Delta_k D^i z \in \Theta^c : z \in \mathbb{Z}_+^{|D^i|}\right\}$. Let $u^c = X_k^c + \Delta_k D^i z_u$ and $v^c = X_k^c + \Delta_k D^i z_v$ where $z_u,\ z_v \in \mathbb{Z}_+^{|D^i|}$. Since $u^d = v^d$ but $u \neq v$, then $u^c \neq v^c$. If follows that $z_u \neq z_v$. Then,

$$\begin{aligned}
\|u^c - v^c\| &= \left\|X_k^c + \Delta_k D^i z_v - X_k^c - \Delta_k D^i z_u\right\| \\
&= \Delta_k \left\|D^i (z_v - z_u)\right\| \\
&= \Delta_k \left\|G_i Z_i (z_v - z_u)\right\| \\
&\geq \Delta_k \frac{\|Z_i(z_v - z_u)\|}{\|G_i^{-1}\|} \\
&\geq \frac{\Delta_k}{\|G_i^{-1}\|} \quad .
\end{aligned}$$

The last inequality holds because $\|Z_i(z_v - z_u)\| \geq 1$; *i.e.*, $Z_i(z_v - z_u)$ is a nonzero integer vector with norm at least 1. ■

Lemma 3.16 *With probability 1, there exists a positive integer $b^u < \infty$ such that $\Delta_k \leq \Delta_0 \tau^{b^u}$ for any $k \geq 0$, $k \in \mathbb{Z}$.*

Proof. By assumption **A1**, the search domain is bounded so the discrete variables can only take on a finite number of values. Let $i_{\max}$ denote this number and let $I = \{1, \ldots, i_{\max}\}$. Also under assumption **A1**, for each $i \in I$, let Λ_i be a compact set in $\mathbb{R}^{n^c}$ containing all MGPS-RS iterates whose discrete variable values correspond to $i \in I$. Let $\gamma = \max_{i\in I} \text{diam}(\Lambda_i)$ and $\beta = \max_{i\in I} \left\|G_i^{-1}\right\|$, where diam$(\cdot)$ denotes the maximum distance between any two points in the set. If $\Delta_k > \gamma\beta$, then by Lemma 3.15 (with $v = X_k$), any mesh point u with $u^c \neq X_k^c$ would be outside of $\bigcup_{i\in I}\Lambda_i$. This can be seen by the following:

$$\begin{aligned} \|u^c - X_k^c\| \quad &\geq \quad \frac{\Delta_k}{\left\|G_i^{-1}\right\|} > \frac{\gamma\beta}{\left\|G_i^{-1}\right\|} = \frac{\gamma \max_{i\in I}\left\|G_i^{-1}\right\|}{\left\|G_i^{-1}\right\|} \geq \gamma = \max_{i\in I} \text{diam}(\Lambda_i) \qquad (3.23) \\ &\Longrightarrow \quad \|u^c - X_k^c\| > \max_{i\in I} \text{diam}(\Lambda_i) \ . \end{aligned}$$

Thus, $\Delta_k > \gamma\beta$ implies that the continuous part of the mesh is devoid of candidates except for the incumbent. Therefore, $M_k(X_k) = \Theta^d \times \{X_k^c\}$ and $P_k(X_k) = \{X_k\}$. Furthermore, the poll set for any discrete neighbor Y of X_k is devoid of candidates except for Y by the same argument as (3.23) using Lemma 3.15 (with $V = Y$), so the EXTENDED POLL step is avoided.

The algorithm can consider a maximum of $i_{\max}$ different candidates defined by the combinations of Θ^d during a SEARCH or POLL step. The mesh size parameter grows without bound only if it is possible to cycle indefinitely between these $i_{\max}$ solutions. But Lemma 3.14 guarantees $P(K_S < \infty) = 1$ where K_S is the number of consecutive successful iterations after iteration k. Then the mesh size parameter will have grown, at a maximum, by a factor of $(\tau^{m_{\max}})^{K_S}$ and is thus bounded above by $\gamma\beta(\tau^{m_{\max}})^{K_S}$. Let b^u be large enough so that $\Delta_0\tau^{b^u} \geq \gamma\beta(\tau^{m_{\max}})^{K_S}$. Then $P(K_S < \infty) = 1 \Longrightarrow P(\gamma\beta(\tau^{m_{\max}})^{K_S} < \infty) = 1 \Longrightarrow P(\Delta_0\tau^{b^u} < \infty) = 1 \Longrightarrow P(b^u < \infty) = 1.$ ■

Theorem 3.17 *The mesh size parameters satisfy* $P\left(\liminf_{k\to+\infty} \Delta_k = 0\right) = 1.$

Proof. By way of contradiction, suppose there exists a negative integer b^ℓ such that $\Delta_0\tau^{b^\ell} > 0$ and $P(\Delta_k > \Delta_0\tau^{b^\ell}) = 1$ for all $k \geq 0$, $k \in \mathbb{Z}$. By definition of the update rules,

Δ_k can be expressed as $\Delta_k = \tau^{b_k}\Delta_0$ for some $b_k \in \mathbb{Z}$ (see (3.13)). Since Lemma 3.16 ensures that b_k is bounded above *a.s.* by b^u, it follows that $b_k \in \{b^\ell, b^\ell+1, \ldots, b^u\}$ *a.s.* Thus, b_k is an element of a finite set of integers which implies that Δ_k takes on a finite number of values for all $k \geq 0$.

Now, $X_{k+1} \in M_k$ ensures that $X_{k+1}^c = X_k^c + \Delta_k D^i z_k$ for some $z_k \in \mathbb{Z}_+^{|D_i|}$ and some $i \in \{1, 2, \ldots, i_{\max}\}$. Repeated application of this equation leads to the following result over a fixed i at iteration $N \geq 1$, where p and q are relatively prime integers satisfying $\tau = \frac{p}{q}$:

$$\begin{aligned}
X_N^c &= X_{N-1}^c + \Delta_{N-1} D^i z_{N-1} \\
&= \left(X_{N-2}^c + \Delta_{N-2} D^i z_{N-2}\right) + \Delta_{N-1} D^i z_{N-1} \\
&= \vdots \\
&= X_0^c + \Delta_0 D^i z_0 + \Delta_1 D^i z_1 + \cdots + \Delta_{N-1} D^i z_{N-1} \\
&= X_0^c + \sum_{k=0}^{N-1} \Delta_k D^i z_k \\
&= X_0^c + D^i \sum_{k=0}^{N-1} \Delta_0 \tau^{b_k} z_k \\
&= X_0^c + \Delta_0 D^i \sum_{k=0}^{N-1} \left(\frac{p}{q}\right)^{b_k} z_k \\
&= X_0^c + \Delta_0 D^i \sum_{k=0}^{N-1} \frac{p^{(b_k + b^\ell - b^\ell)}}{q^{(b_k + b^u - b^u)}} z_k \\
&= X_0^c + \frac{p^{b^\ell}}{q^{b^u}} \Delta_0 D^i \sum_{k=0}^{N-1} \frac{p^{(b_k - b^\ell)}}{q^{(b_k - b^u)}} z_k \\
&= X_0^c + \frac{p^{b^\ell}}{q^{b^u}} \Delta_0 D^i \sum_{k=0}^{N-1} p^{(b_k - b^\ell)} q^{(b^u - b_k)} z_k
\end{aligned}$$

Since $p^{(b_k - b^\ell)}$ and $q^{(b^u - b_k)}$ are both integers, then $\sum_{k=0}^{N-1} p^{(b_k - b^\ell)} q^{(b^u - b_k)} z_k$ is a $|D^i|$-dimensional vector of integers (recall $z_k \in \mathbb{Z}_+^{|D^i|}$). So, the continuous part of each iterate, X_k^c, $k = 0, \ldots, N$ having the same discrete variable values defined by i lies on the translated integer lattice generated by X_0^c and the columns of $\frac{p^{b^\ell}}{q^{b^u}}\Delta_0 D^i$. Furthermore, the discrete part of each iterate, X_k^d, lies on the integer lattice $\Theta^d \subset \mathbb{Z}^{n^d}$. By assumption **A1**, all iterates belong to a compact set, so there must be only a finite number of possible iterates.

Lemma 3.14 ensures that the algorithm cannot cycle indefinitely between these points (*i.e.* the subsequence of consecutive successful iterations is finite *a.s.*). Thus, as $k \to +\infty$, one of the iterates must be visited infinitely many times *a.s.*, which implies an infinite number of mesh refinements. But this contradicts the hypothesis that $P(\Delta_k > \Delta_0 \tau^{b^\ell}) = 1$ as $k \to +\infty$. Therefore, $P(\Delta_k > \Delta_0 \tau^{b^\ell}) = 0$, which implies $P\left(\liminf_{k\to+\infty} \Delta_k = 0\right) = 1$. ■

The results of this section illustrate the importance of the restriction that $D^i = G_i Z_i$ (Equation (3.7)). Under assumption **A1**, this ensures that the mesh has a finite number of points in Θ. This, combined with the "finiteness" of incorrectly selected iterates, ensures that there can only be a finite number of consecutive successful iterations.

3.7.3 Main Results

In this section, the existence of limit points for MGPS-RS iterates is proven. In addition, limit points are shown to satisfy the first-order necessary conditions for optimality in Definition 3.2. The results have been modified from [13] and [1] to accommodate the new algorithmic framework. The following definition, which distinguishes a subsequence of the unsuccessful iterates, simplifies the analysis.

Definition 3.18 (Refining subsequence) A subsequence of unsuccessful MGPS-RS iterates $\{X_k\}_{k\in K}$ (for some subset of indices K) is said to be a *refining subsequence* if $\{\Delta_k\}_{k\in K}$ converges almost surely to zero, *i.e.*, $P\left(\lim_{k\in K} \Delta_k = 0\right) = 1$.

Since Δ_k shrinks for unsuccessful iterations, Theorem 3.17 guarantees that the MGPS-RS algorithm has, with probability 1, infinitely many such iterations. The next theorem, similar to the results from [13] and [1] but modified here for the probabilistic setting, establishes the existence of certain limit points associated with refining subsequences.

Theorem 3.19 *There exists a point $\hat{x} \in \Theta$ and a refining subsequence $\{X_k\}_{k\in K}$, with associated index set $K \subset \{k : X_{k+1} = X_k\}$ such that $\{X_k\}_{k\in K}$ converges almost surely to $\hat{x}$. Moreover, if $\mathcal{N}$ is continuous at $\hat{x}$, then there exists $\hat{y} \in \mathcal{N}(\hat{x})$ and $\hat{z} = (\hat{z}^c, \hat{y}^d) \in \Theta$ such that $\{Y_k\}_{k\in K}$ converges almost surely to $\hat{y}$ and $\{Z_k\}_{k\in K}$ converges almost surely to $\hat{z}$ where each $Z_k \in \Theta$ is an* EXTENDED POLL *endpoint initiated at $Y_k^0 \in \mathcal{N}(X_k)$.*

Proof. Theorem 3.17 guarantees $P\left(\liminf_{k\to+\infty} \Delta_k = 0\right) = 1$; thus there is an infinite subset of indices of unsuccessful iterates $K' \subset \{k : X_{k+1} = X_k\}$, such that the subsequence $\{\Delta_k\}_{k\in K'}$ converges *a.s.* to zero, *i.e.*, $P\left(\lim_{k\in K'} \Delta_k = 0\right) = 1$. Since all iterates X_k lie in a compact set, there exists an infinite subset of indices $K'' \subset K'$ such that the subsequence $\{X_k\}_{k\in K''}$ converges almost surely. Let $\hat{x}$ be the limit point of such a subsequence.

The continuity of $\mathcal{N}$ at $\hat{x}$ guarantees that $\hat{y} \in \mathcal{N}(\hat{x}) \subset \Theta$ is a limit point of a subsequence $Y_k \in \mathcal{N}(X_k)$. Let $\hat{z} \in \Theta$ be a limit point of the sequence $Z_k \in \Theta$ of EXTENDED POLL endpoints initiated at Y_k^0. Choose $K \subset K''$ to be such that both $\{Y_k\}_{k\in K}$ converges *a.s.* to $\hat{y}$ and $\{Z_k\}_{k\in K}$ converges *a.s.*, letting $\hat{z}$ denote the limit point. ■

For the remainder of the analysis, it is assumed that $\hat{x}$ and K satisfy the conditions of Theorem 3.19. The following lemma establishes the first main result, showing that limit points satisfy necessary condition 2 of Definition 3.2. The direct proof is modified for the stochastic case from [1], where it was presented as an alternative to the contradictory proof in [13].

Lemma 3.20 *If $\mathcal{N}$ is continuous at the limit point $\hat{x}$, then $\hat{x}$ satisfies $f(\hat{x}) \leq f(\hat{y})$* a.s. *for all $\hat{y} \in \mathcal{N}(\hat{x})$.*

Proof. From Theorem 3.19, the sequences $\{X_k\}_{k\in K}$ and $\{Y_k\}_{k\in K}$ converge *a.s.* to $\hat{x}$ and $\hat{y}$, respectively. Since $k \in K \subset \{k : X_{k+1} = X_k\}$, each $\{X_k\}_{k\in K}$ meets one of the conditions (3.20)–(3.22). Assumption **A8** ensures that the number of iterates satisfying condition

(3.22) is finite. Furthermore, since the set iterates meeting condition (3.21) is a subset of all incorrectly selected iterates, Lemma 3.13 ensures the number of iterates satisfying this condition is finite almost surely. Therefore, the number of correctly selected iterates in $\{X_k\}_{k\in K}$ meeting condition (3.20) must be infinite. Let k' denote an unsuccessful iteration after the last occurrence of both conditions (3.21) and (3.22) and let $K' = K \cap \{k \geq k'\}$ which converges *a.s.* to $\hat{x}$. Since each iterate $\{X_k\}_{k\in K'}$ meets condition (3.20), $f(X_k) < f(Y_k)$ for all $k \in K'$. By the continuity of $\mathcal{N}$ and assumption **A2**, $f(\hat{x}) = \lim_{k\in K'} f(X_k) \leq \lim_{k\in K'} f(Y_k) = f(\hat{y})$. ■

The following lemma is necessary to show stationarity of the iterates X_k, and EXTENDED POLL endpoints Z_k. It merges two lemmas from [13] and modifies the results therein for the new algorithmic framework.

Lemma 3.21 *Let $\hat{w}$ be the limit point of a refining subsequence $\{W_k\}_{k\in K}$. Then $(w^c - \hat{w}^c)^T \nabla^c f(\hat{w}) \geq 0$* a.s. *for any feasible $(w^c, \hat{w}^d)$.*

Proof. By assumption **A2**, the mean value theorem applies, *i.e.*, for points x_1 and x_2 satisfying $x_1^d = x_2^d$,

$$f(x_2) = f(x_1) + (x_2^c - x_1^c)^T \nabla^c f(x) \quad \text{where } x^c \in [x_1^c, x_2^c] \ .$$

For $x_1 = W_k$, $x_2 = V = W_k + \Delta_k(d,0) \in P_k(W_k)$, and any $d \in D_k^i \subseteq D^i$ that is feasible infinitely often, substitution yields

$$f(V) = f(W_k + \Delta_k(d,0)) = f(W_k) + \Delta_k d^T \nabla^c f(W_k + \lambda_k^d \Delta_k(d,0)) \tag{3.24}$$

for $\lambda_k^d \in [0,1]$ that depends on the iteration k and positive basis vector d. Choose $k \in K$ large enough so that the indifference zone condition is satisfied and incorrect selections have terminated almost surely. Then by condition (3.20), $f(V) - f(W_k) \geq \delta_r(k)$ where $\delta_r(k)$

depends on k. Furthermore,

$$\begin{aligned}
f(W_k) &\leq \min_{V\in P(W_k)} f(V) - \delta_r(k) \\
&= \min_{d\in D_k^i} \left\{ f(W_k) + \Delta_k d^T \nabla^c f(W_k + \lambda_k^d \Delta_k(d,0)) \right\} - \delta_r(k) \\
&= f(W_k) - \delta_r(k) + \Delta_k \min_{d\in D_k^i} \left\{ d^T \nabla^c f(W_k + \lambda_k^d \Delta_k(d,0)) \right\} ,
\end{aligned}$$

which implies that

$$\min_{d\in D_k^i} \left\{ d^T \nabla^c f(W_k + \lambda_k^d \Delta_k(d,0)) \right\} \geq \delta_k \ .$$

Taking the limit as $k \to \infty$ (in K) yields $\min_{d\in D^i} \left\{ d^T \nabla^c f(\hat{w}) \right\} \geq 0$ *a.s.* (since $\lim_{k\to\infty} \delta_r(k) = 0$ by assumption **A6**). Therefore, $d^T \nabla^c f(\hat{w}) \geq 0$ *a.s.* for any $d \in D^i$ that is feasible infinitely often.

By assumption **A4**, any feasible direction $(w^c - \hat{w}^c)$ is a nonnegative linear combination of feasible directions in D^i that span the tangent cone of Θ^c at $\hat{w}$. Then for $\beta_j \geq 0$, $j = 1, 2, \ldots, n_d$, $(w^c - \hat{w}^c) = \sum_{i=1}^{n_d} \beta_j d_j$ and

$$(w^c - \hat{w}^c)^T \nabla^c f(\hat{w}^c) = \sum_{j=1}^{n_d} \beta_j d_j^T \nabla^c f(\hat{w}^c) \geq 0 \quad \textit{a.s.}. \qquad \blacksquare$$

It is now possible to state the second main result. Lemma 3.22 shows that the limit point $\hat{x}$ satisfies condition 1 of Definition 3.2.

Lemma 3.22 *The limit point* $\hat{x}$ *satisfies* $(x^c - \hat{x}^c)^T \nabla^c f(\hat{x}) \geq 0$ a.s. *for any feasible* $(x^c, \hat{x}^d)$.

Proof. The result follows directly from Lemma 3.21 by substituting X_k for W_k as the refining subsequence, and from results on the sequence $\{X_k\}_{k\in K}$ of Theorem 3.19. $\blacksquare$

The remaining result may now be completed. Lemma 3.23 shows that limit points $\hat{x}$ and discrete neighbors $\hat{y}$ that satisfy $f(\hat{y}) = f(\hat{x})$ meet condition 3 of Definition 3.2. Theorem 3.24 collects all the main results into a single theorem.

Lemma 3.23 *The limit point $\hat{x}$ and any point $\hat{y}$ in the set of neighbors $\mathcal{N}(\hat{x})$ satisfying $f(\hat{y}) = f(\hat{x})$, are such that $(y^c - \hat{y}^c)^T \nabla^c f(\hat{y}) \geq 0$ a.s. for any feasible $(y^c, \hat{y}^d)$.*

Proof. Choose $k' \in K$ large enough so that the indifference zone condition is satisfied and incorrect selections have terminated almost surely and let $K' = K \cap \{k \geq k'\}$. Then by condition (3.17), $f(Y_k^j) < f(Y_k^{j-1})$ for all $k \in K'$, which implies $f(Z_k) < f(Y_k)$ for all $k \in K'$. Furthermore, since K' is a subset of unsuccessful iterates, condition (3.20) is satisfied, which implies $f(X_k) < f(Z_k)$ for each $k \in K'$. By continuity of f and taking the limit as $k \to \infty$ (in K'), it follows that $f(\hat{x}) \leq f(\hat{z}) \leq f(\hat{y})$. Therefore, $f(\hat{z}) = f(\hat{y})$.

By the differentiability of f, it follows that

$$\begin{aligned}
(y^c - \hat{y}^c)^T \nabla^c f(\hat{y}) = f'(\hat{y}; (y^c - \hat{y}^c, 0)) &= \lim_{t \to 0} \frac{f(\hat{y} + t(y^c - \hat{y}^c, 0)) - f(\hat{y})}{t} \\
&= \lim_{k \in K'} \frac{f(Y_k) - f(\hat{y})}{\Delta_k} = \lim_{k \in K'} \frac{f(Y_k) - f(\hat{z})}{\Delta_k} \geq \lim_{k \in K'} \frac{f(Z_k) - f(\hat{z})}{\Delta_k} \\
&= (z^c - \hat{z}^c)^T \nabla^c f(\hat{z}) \geq 0,
\end{aligned}$$

where $f'(\hat{y}; (y^c - \hat{y}^c, 0))$ denotes the directional derivative of f at $\hat{y}$ in the direction $(y^c - \hat{y}^c, 0)$, and the last inequality follows by substituting Z_k for W_k as the refining subsequence in Lemma 3.21. ■

Theorem 3.24 *The limit point $\hat{x}$ satisfies first-order necessary conditions for optimality* a.s..

Proof. The result, based on conditions 1–3 of Definition 3.2, follows directly from Lemmas 3.20, 3.22, and 3.23. ■

3.8 Illustrative Example

Prior to a comprehensive computational evaluation of specific algorithm implementations in Chapter 5, a basic version of the algorithm is illustrated on a small unconstrained problem with two continuous variables and one discrete (binary) variable. Consider the response function

$$F(x) = f(x) + N(0, \sigma^2(f(x)) \tag{3.25}$$

where $N(0, \sigma^2(f(x))$ is a normally distributed, mean-zero noise term added to an underlying true objective function. The variance σ^2 of the noise depends on the true function $f(x)$, which is defined over $(x_1, x_2)^T \in \mathbb{R}^2$ and $x_3 \in \{0, 1\}$ as

$$f(x) = f_1(x_1, x_2)(1 - x_3) + f_2(x_1, x_2)x_3 \tag{3.26}$$

where the functions f_1 and f_2 are overlapping quadratic functions (see Figure 3.5) as follows:

$$\begin{aligned} f_1(x_1, x_2) &= (x_1 - 9/4)^2 + (x_2 - 9/4)^2 + 1, \\ f_2(x_1, x_2) &= 1/2(x_1 - 3/2)^2 + 1/2(x_2 - 3/2)^2 + 7/4. \end{aligned}$$

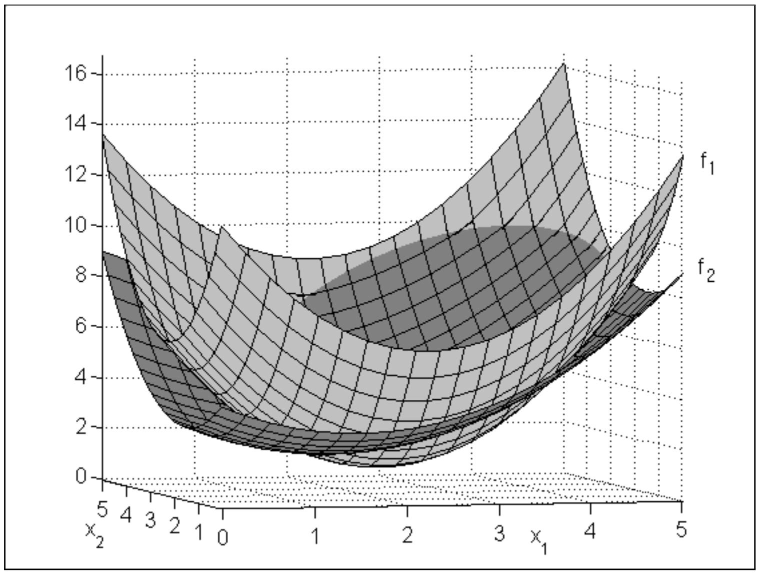
The optimum is located at $x^* = (x_1^*, x_2^*, x_3^*) = (\frac{9}{4}, \frac{9}{4}, 0)$ with $f(x^*) = 1$.

To compare two different random noise scenarios, the standard deviation of the error term $\sigma(f(x))$ is either proportional or inversely proportional to f:

$$\begin{aligned} \sigma_1(f(x)) &= f(x), \quad \text{or} \\ \sigma_2(f(x)) &= \frac{1}{f(x)}. \end{aligned}$$

These test cases are referred to as noise cases 1 and 2, respectively. At optimality, σ_1 and σ_2 are equal but diverge for trial points away from optimality.

The two-stage indifference-zone procedure of Rinott for unequal variances [114] was implemented as the R&S method. This procedure uses two stages of sampling to estimate the true mean of the response function for each candidate. In the first stage, the sample variance S_q^2 for each candidate q is computed from a fixed number of response samples for each candidate. This information is used to determine the number of second-stage samples



$f(x_1, x_2, x_3) = f_1(x_1, x_2)(1 - x_3) + f_2(x_1, x_2)x_3$

Figure 3.5. **Example Test Function.**

required to guarantee a probability of correct selection. Given s_0 first-stage samples with sample variance S_q^2 for each candidate q, Rinott's procedure prescribes $s_q - s_0$ additional samples, where

$$s_q = \max\left\{ s_0, \left\lceil \left(\frac{gS_q}{\delta} \right)^2 \right\rceil \right\}, \tag{3.27}$$

$g = g(n_C, \alpha, s_0)$ is *Rinott's constant*, and $\lceil m \rceil$ indicates the smallest integer greater than or equal to m (ceiling function). Tabulated values for g have been published for commonly used parameter combinations but was computed numerically in this investigation by adapting the code listed in [27] to accommodate changing parameter settings. The objective function value is then estimated by averaging the response samples over both stages for each candidate. To satisfy the requirements on the R&S parameters, both δ_r and α_r were decremented geometrically, *i.e.* $\delta_r = \delta_0(\rho_\delta)^r$ and $\alpha_r = \alpha_0(\rho_\alpha)^r$. The following settings were used in the numerical experiment: $s_0 = 5$, $\delta_0 = 1.0$, $\alpha_0 = 0.4$, and $\rho_\delta = \rho_\alpha = .95$.

For this example, the algorithm was implemented with an empty SEARCH step. The direction set consisted of the coordinate axes, *i.e.* $D_k^i = [I, -I]$ for all k and both settings of i. The discrete neighbor set was defined as $\mathcal{N}(x) = \{(x_1, x_2, x_3), (x_1, x_2, 1 - x_3)\}$. The step size parameters were set to $\tau = \frac{9}{8}$, $m_k^- = -2$ for all k, and $m_k^+ = 1$ for all k so that $\Delta_{k+1} = (\frac{8}{9})^2 \Delta_k$ for refinement and $\Delta_{k+1} = \frac{9}{8}\Delta_k$ for coarsening. These parameters were selected so that the search steps lengthened after successful iterations but not so much as to cause high variance for candidates in the poll set when near the optimal solution. The initial step size was set to $\Delta_0 = 0.5$. The extended poll trigger $\xi_k = \frac{3}{4}$ was used for all k to ensure that, at the minimum $\bar{x}^c = (\bar{x}_1, \bar{x}_2) = (\frac{3}{2}, \frac{3}{2})$ of f_2, the surface of f_1 is polled since $f_1(\bar{x}^c) - f_2(\bar{x}^c) = \frac{3}{8}$.

In the numerical experiment, the algorithm was replicated twenty times for each noise case. All forty replications were initiated from starting solution $X_0 = (0, 5, 1)$, $f(X_0) = 9$. For each noise case, the following metrics were used to gauge the performance:

- average number of candidate solutions visited,
- average distance of terminal solution from x^*: $\|x^t - x^*\|$,
- average difference of terminal solution in true objective function value from $f(x^*)$: $|f^t - f^*|$, and
- average number of iterations completed.

The distance from optimum was measured as the sum of the Euclidean distance in the continuous domain and the value of the discrete variable $\|x^t - x^*\| = \|(x^c)^t - (x^c)^*\| + (x^d)^t$. Each of the metrics was recorded at ten predetermined stages of algorithm progression, measured in terms of the number of responses sampled, and averaged over the twenty replications. The results for noise cases 1 and 2 are presented in Tables 3.1 and 3.2, respectively.

Table 3.1. **MGPS-RS Average Performance for Noise Case 1 over 20 Replications.**

Response Samples	Candidates Visited	$\|x^t - x^*\|$	$\|f^t - f^*\|$	Iterations
0	-	4.55	8.00	-
2,500	.9	4.50	7.76	.2
5,000	6.9	4.13	6.26	1.2
7,500	13.5	3.67	4.53	2.5
10,000	22.8	2.85	2.82	4.4
20,000	73.5	.657	.343	13.4
30,000	99.8	.587	.294	17.9
40,000	116.2	.449	.198	20.9
50,000	130.1	.464	.185	23.5
75,000	150.6	.350	.137	27.6
100,000	166.4	.279	.122	30.8

Table 3.2. **MGPS-RS Average Performance for Noise Case 2 over 20 Replications.**

Response Samples	Candidates Visited	$\|x^t - x^*\|$	$\|f^t - f^*\|$	Iterations
0	-	4.55	8.00	-
2,500	113.7	.495	.272	19.9
5,000	133.0	.411	.204	23.1
7,500	145.5	.382	.184	25.3
10,000	153.3	.376	.172	26.8
20,000	173.0	.331	.143	30.6
30,000	185.0	.302	.123	32.9
40,000	193.0	.288	.113	34.5
50,000	200.5	.264	.095	36.0
75,000	213.0	.234	.073	38.5
100,000	222.0	.219	.062	40.3

The results clearly illustrate the effects of the response variance on algorithm performance. Since $\sigma_2 = \frac{1}{81}\sigma_1$ at the starting solution, the algorithm in noise case 2 is able to reach better solutions much more rapidly than in noise case 1. In fact, it takes the algorithm approximately 40,000 response samples in case 1 to reach equivalent progress achieved after 2,500 samples in case 2. After 100,000 response samples, the algorithm in case 2 outperforms case 1 by approximately 30% in the number of candidate solutions considered and number of iterations completed while finding a solution roughly twice as good in terms of quality of the true objective function value. On the other hand, after achieving significant progress after 2,500 response samples in case 2, progress slows considerably after reaching a region of the search space where the standard deviation of the noise approaches unity and the surface begins to flatten. For noise case 1, the error control measures built into the algorithm enable consistent progress despite the challenging situation introduced by high response variation. It should be noted that response variation has a profound effect on computational requirements relative to the deterministic case. By comparison, applying the algorithm to the noise-free version of this problem (*i.e.* $\sigma = 0$) produced a solution that was within 0.0134 of x^* with an objective function value within 0.000181 of $f(x^*)$ after 300 function samples.

For both noise cases, the solution found after 100,000 response samples was on the surface of f_1. In noise case 1, sixteen of the twenty replications had permanently reached the surface of f_1 by 20,000 response samples, which accounts for the significant improvement between 10,000 and 20,000 samples in Table 3.1. In noise case 2, all iterates after the eleventh iteration, on average, remained on f_1, well before 2,500 responses were evaluated. In the most extreme case under noise case 2, the maximum number of iterations required before all iterates remained on f_1 was twenty-three. In this case, the algorithm found a point in the continuous design space for which the values of f_1 and f_2 were very close in magnitude. In particular, at the point $\tilde{x}^c = (1.645, 1.692)$, the value $|f_1(\tilde{x}^c) - f_2(\tilde{x}^c)| = 0.111$. The algorithm alternated between discrete neighbors $(\tilde{x}^c, 0)$ and $(\tilde{x}^c, 1)$ from iteration 13 until iteration 23, during which time approximately 2,000 response samples were obtained and the number of R&S procedures performed by the algorithm increased from 18 to 37. As a result, the indifference zone parameter had been reduced from $\delta_{18} = (.95)^{18} = 0.397 > |f_1(\tilde{x}^c) -$

$f_2(\tilde{x}^c)|$ to $\delta_{37} = (.95)^{37} = 0.150 > |f_1(\tilde{x}^c) - f_2(\tilde{x}^c)|$. Therefore, the R&S procedure could not prescribe enough samples to detect the best solution among the two discrete neighbors until δ_r had been reduced to a value that approached the absolute difference between the two neighbors. This isolated case illustrates the potential computational requirements necessary to detect small differences between candidate solutions in the presence of random variation.

To illustrate the asymptotic behavior of the algorithm, the algorithm was run again starting from the optimal point $X_0 = x^* = (\frac{9}{4}, \frac{9}{4}, 0)$ with the standard deviation of the noise term $\sigma = 2$ throughout the design space. For this run, the same parameter settings as in the original experiment were used except for the initial significance level, which was set to $\alpha_0 = 0.8$ to encourage erroneous iterate selections for the purpose of illustration. The run was terminated after two million response samples. The iteration history is depicted in Figure 3.6. The figure also plots the decay of the indifference zone parameter as the downward sloping curve as well as the cumulative response samples as the upward sloping dashed line (with scale on the right).

The plot shows that, although starting from the optimal point, many unimproving steps are taken, indicated by an increase in true objective value of the iterates. However, the magnitude of the difference between successive iterates, in most cases, is within the tolerance defined by the indifference zone line. Three exceptions occur at iterations 1, 8, and 13, when the true function value for the iterates jumps above the indifference zone boundary[2]. In these three cases, the significance levels were $\alpha_1 = .8(.95) = 0.76$, $\alpha_8 = .8(.95)^8 = 0.53$, and $\alpha_{13} = .8(.95)^{13} = 0.41$, respectively. (Note that no extended poll steps were performed so $r = k$ throughout the iteration sequence.) Therefore, the three iterates selected at iterations 1, 8, and 13 represent incorrect selections for which the probability of incorrect selection

[2]Iteration 4, as well as iterations 36 through 40, are not examples of these exceptions, even though the objective function values exceed $f(x^*) + \delta_k$, because the differences between $f(x_k)$ at these iterations and that of the previous iterate do not exceed δ_k.

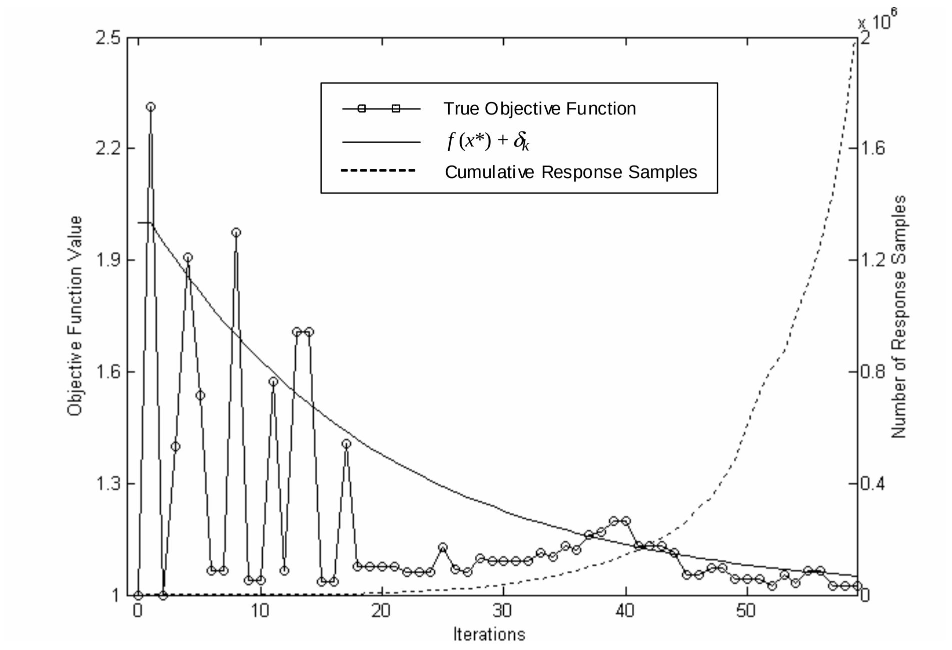


Figure 3.6. **Asymptotic behavior of MGPS-RS, shown after 2 million response samples.**

was 0.76, 0.53 and 0.41, respectively. However, as the iteration sequence continues, the search settles down near the optimal point and the magnitude of unimproving solutions decreases commensurate with the indifference zone parameter. Also, no additional iterates are incorrectly selected since the significance level (not shown in the plot) is decaying at the same rate as the indifference zone parameter. However, the costs associated with error control during the latter stages of the search are evident with the rapid increase in response samples required per iteration.

Figure 3.6 illustrates the computational implications of achieving better solutions as the search progresses. Clearly, for MGPS-RS algorithms to have practical value, it is important to address concerns regarding the sampling effort required. In this chapter, the mathematical framework of MGPS-RS was presented and its convergence properties rigorously established. In the following chapter, various implementation alternatives are described that seek efficient use of the sampling budget.

Chapter 4 - Algorithm Implementations

In this chapter, the details of the various MGPS-RS algorithm implementations are presented. Particular attention is given to implementations that have the potential to provide computational enhancements to the basic algorithm. Two essential ideas are presented that specifically address methods to improve the computational performance of the algorithms. The first, described in Section 4.1, is the use of modern ranking and selection techniques to offer more efficient sampling strategies relative to the basic procedure of Rinott. The second idea is to augment the search by using surrogate functions during the SEARCH step as a means to model the relationship between input designs and response outputs based on previously obtained samples. The goal of this approach, introduced in Section 4.2, is to develop an inexpensive method to nominate high quality trial points and thus accelerate algorithm convergence. Another important concept relevant to computational performance is discussed in Section 4.3, which proposes a strategy for establishing appropriate algorithm termination criteria. The strategy seeks to avoid additional sampling when further sampling would lead to marginal returns on objective function value improvement. Section 4.4 unifies the various implementation considerations into an overarching algorithm design that describes implementation in further detail for the algorithm substeps and summarizes the algorithm parameters. Section 4.5 summarizes the key points of the chapter.

4.1 Specific Ranking and Selection (R&S) Procedures

An important concern for implementation is the selection of specific R&S procedures. Since unknown and unequal variances are allowed, a procedure having at least two stages is required, allowing the sample variance to be computed in an initial stage. Three such procedures were selected for implementation and computational evaluation: Rinott's two-stage procedure [114], a screen-and-select (SAS) procedure of Nelson *et al.* [99], and the Sequential Selection with Memory (SSM) procedure of Pichitlamken and Nelson [109].

Rinott's two-stage procedure, described in Section 3.8, is a well-known, simple procedure that satisfies the probability of correct selection guarantee (3.15). It uses the sample variance from a fixed number of first-stage samples for each candidate to determine the number of second-stage samples required to guarantee the probability of correct selection. A detailed listing of Rinott's procedure, adapted from [27, p.61], is provided in Figure 4.1. In the procedure, the number of second-stage samples is dependent on *Rinott's constant* $g = g(n_C, \alpha, \nu)$, which is the solution to the equation

$$\int_0^\infty \left[\int_0^\infty \Phi\left(\frac{g}{\sqrt{\nu(1/x + 1/y)}} \right) f_\upsilon(x)dx \right]^{n_C - 1} f_\upsilon(y)dy = 1 - \alpha \qquad (4.1)$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function and $f_\upsilon(\cdot)$ is the probability distribution function of the χ^2-distribution with υ degrees of freedom. This constant can be obtained from a table of values or computed numerically. To account for the changing parameter α in the computational evaluation of Chapter 5, a MATLAB$^{®}$ m-file was written to compute g that was based on the FORTRAN program `RINOTT` listed in Appendix C of [27].

Rinott's procedure can be computationally inefficient because it is constructed based on the *least favorable configuration* assumption that the best candidate has a true mean exactly δ better than all remaining candidates, which are all tied for second best [140]. As a result, the procedure can overprescribe the number of required second stage samples in order to guarantee the $P\{CS\}$. Furthermore, the procedure has no mechanism to consider the sample mean of the responses after the first stage, and therefore cannot eliminate clearly inferior candidates prior to conducting additional sampling. These characteristics

For a candidate set C indexed by $q \in \{1, \ldots, n_C\}$, fix the common number of replications $s_0 \geq 2$ to be taken in Stage 1, significance level α, and indifference zone parameter δ. Find the constant $g = g(n_C, \alpha, \nu)$ that solves (4.1) where $\nu = s_0 - 1$.

Stage 1: For each candidate q, collect s_0 response samples F_{qs}, $s = 1, \ldots, s_0$.

Stage 2: Calculate the sample means and variances based on the stage 1 samples, $\bar{F}_q(s_0) = s_0^{-1} \sum_{s=1}^{s_0} F_{qs}$ and $S_q^2 = \nu^{-1} \sum_{s=1}^{s_0} (F_{qs} - \bar{F}_q(s_0))$. Collect $s_q - s_0$ additional response samples for candidate $q = 1, 2, \ldots, n_C$ where

$$s_q = \max \left\{ s_0, \left\lceil (g S_q/\delta)^2 \right\rceil \right\} \ .$$

Calculate $\bar{F}_q(s_q) = s_q^{-1} \sum_{s=1}^{s_q} F_{qs}$, $q = 1, 2, \ldots, n_C$ based on the combined results of the Stage 1 and Stage 2 samples. Select the candidate associated with the smallest sample mean over both stages, $\min_q \{\bar{F}_q(s_q)\}$, as having the δ-near-best mean.

Figure 4.1. **Rinott Selection Procedure (adapted from [27])**

are especially problematic within an iterative search framework since the R&S procedure is executed repeatedly and the number of unnecessary samples accumulates at each iteration, limiting the progress of the algorithm relative to a fixed budget of response samples.

The SAS procedure alleviates some of the computational concerns of Rinott's procedure by combining Rinott's procedure with a screening step that can eliminate some solutions after the first stage. For an overall significance level α, significance levels α_1 and α_2 are chosen for screening and selection, respectively, such that $\alpha = \alpha_1 + \alpha_2$. After collecting s_0 samples of each candidate in the first stage, those candidates with a sample mean that is significantly inferior to the best of the rest are eliminated from further sampling. The set of surviving candidates is guaranteed to contain the best with probability at least $1 - \alpha_1$ as long as the indifference zone condition is met. Then, $s_q - s_0$ second stage samples are required only for the survivors according to (3.27) except at significance level α_2 instead of α. Nelson *et al.* [99] prove that the combined procedure satisfies (3.15). A detailed listing of the combined screen-and-select procedure is provided in Figure 4.2.

For a candidate set C indexed by $q \in \{1, \dots, n_C\}$, fix the common number of first-stage samples $s_0 \geq 2$, overall significance level $\alpha = \alpha_1 + \alpha_2$, screening significance level α_1, selection significance level α_2, and indifference zone parameter δ. Set $t = t_{(1-\alpha_1)^{\frac{1}{n_C - 1}}, \nu}$ and $g = g(n_C, \alpha_2, \nu)$, where $t_{\beta,\nu}$ is the β quantile of the t distribution with $\nu = s_0 - 1$ degrees of freedom and g is Rinott's constant that solves (4.1).

Stage 1. (Screening) For each candidate $q = 1, 2, \dots n_C$, collect s_0 response samples F_{qs}, $s = 1, \dots, s_0$. Calculate the sample means and variances based on the initial s_0 samples, $\bar{F}_q(s_0) = \sum_{s=1}^{s_0} F_{qs}/s_0$ and $S_q^2 = \nu^{-1} \sum_{s=1}^{s_0} (F_{qs} - \bar{F}_q(s_0))^2$. Let

$$W_{qp} = t \left(\frac{S_q^2}{s_0} + \frac{S_p^2}{s_0} \right)^{1/2} \quad \text{for all } q \neq p.$$

Set $Q = \left\{ q : 1 \leq q \leq n_C \text{ and } \bar{F}_q(s_0) \leq \bar{F}_p(s_0) + (W_{qp} - \delta)^+, \forall p \neq q \right\}$ where $y^+ = \max\{0, y\}$. If $|Q| = 1$, then stop and report the only survivor as the best; otherwise, for each $q \in Q$ compute the second stage sample size

$$s_q = \max \left\{ s_0, \left\lceil (g S_q / \delta)^2 \right\rceil \right\} .$$

Stage 2. (Selection) Collect $s_q - s_0$ additional response samples for the survivors of the screening step $q \in Q$ and compute overall sample means $\bar{F}_q(s_q) = s_q^{-1} \sum_{s=1}^{s_q} F_{qs}$, $q \in Q$. Select the candidate associated with the smallest sample mean over both stages, $\min_q \{\bar{F}_q(s_q)\}$, as having the δ-near-best mean.

Figure 4.2. **Combined Screening and Selection (SAS) Procedure (adapted from [99])**

The SSM procedure extends the notion of intermediate elimination of inferior solutions. It is a *fully sequential* procedure specifically designed for iterative search routines. A fully sequential procedure is one that takes one sample at a time from every candidate still in play and eliminates clearly inferior ones as soon as their inferiority is apparent. The SSM procedure is an extension of the procedure presented in [69]; the difference being that SSM allows the re-use of previously sampled responses when design points are revisited where the procedure in [69] does not.

In SSM, an initial stage of sampling is conducted to estimate the variances between each pair of candidates, indexed by $q, p \in \{1, \dots, n_C\}$, according to,

$$S_{qp}^2 = \frac{1}{\nu} \sum_{s=1}^{s_0} (F_{qs} - F_{ps} - [\bar{F}_q(s_0) - \bar{F}_p(s_0)])^2. \tag{4.2}$$

This is followed by a sequence of screening steps that eliminates candidates whose cumulative sums exceed the best of the rest plus a tolerance level that depends on the variances and parameters δ and α. Between each successive screening step, one additional sample is taken from each survivor and the tolerance level decreases. The procedure terminates when only one survivor remains or after exceeding a maximum number of samples determined after the initial stage. In the latter case, the survivor with the minimum sample mean is selected as the best. Pichitlamken [108] proves that SSM satisfies (3.15). An advantage of this method is that the re-use of previously sampled responses can lead to further computational savings. A detailed listing of SSM is shown in Figure 4.3.

The latter two R&S procedures were implemented because they offer more efficient sampling methods relative to Rinott's procedure when the least favorable configuration assumption does not hold; however, this advantage does not come without cost. In order to reduce sampling, they must repeatedly switch among the various candidates. If each candidate represents a single instance of a simulation model, then there may be a sizable *switching cost* that can require, for example, storing the state information of the current model, saving relevant output data, replacing the executable code in active memory with the code of the next model, and restoring the state information of the next model [59]. An important element of evaluating the R&S procedures in the MGPS-RS framework is to consider the number of cumulative switches required, where the term *switch* denotes each time the algorithm must return to a previously sampled candidate for further sampling during the same iteration.

Rinott's procedure incurs no switches because the second stage of sampling for each candidate can begin immediately after the first stage since the number of second stage samples does not depend on comparisons of output data between candidates. The SAS

Step 1. *Initialization.* For a candidate set C indexed by $q, p \in \{1, \ldots, n_C\}$, fix the common number of minimum samples $s_0 \geq 2$, significance level α, and indifference zone parameter δ. Let V denote the set of solutions visited previously. Let $V^c \subseteq C$ denote the set of solutions seen for the first time. For each $Y_q \in V^c$, collect s_0 response samples F_{qs}, $s = 1, \ldots, s_0$. For each $Y_q \in V \cup C$ with s_q stored responses, collect additional response samples F_{qs}, $s = s_q, s_q + 1, \ldots, s_0$ and set $s_q = s_0$. Update $V^c = V^c \cup Y_q$ and $V = V \backslash Y_q$. Compute variance S^2_{qp} using (4.2) where $\nu = s_0 - 1$.

Step 2. *Procedure parameters*: Let

$$a_{qp} = \frac{\nu S^2_{qp}}{2\delta} \left[\left(\frac{n_C - 1}{2\alpha} \right)^{2/\nu} - 1 \right] . \tag{4.3}$$

Let $R_{qp} = \left\lfloor \frac{2a_{qp}}{\delta} \right\rfloor$, $R_q = \max\limits_{q \neq p}\{R_{qp}\}$, and $R = \max\limits_{q}\{R_q\}$. If $s_0 > R$, then stop and select the solution with the lowest $\bar{F}_q(s_0) = s_0^{-1} \sum_{s=1}^{s_0} F_{qs}$ as the best. Otherwise, let $Q = \{1, \ldots, n_C\}$ be the set of surviving solutions, set $t = s_0$ and proceed to Step 3. From here on V represents the set of solutions for which more than t observations have been obtained, while V^c is the set of solutions with exactly t observations.

Step 3. *Screening*: Set $Q^{\text{old}} = Q$. Let

$$Q = \left\{ q : q \in Q^{\text{old}} \;\text{ and }\; T_q \leq \min_{q \in Q^{\text{old}}, q \neq p} \{T_p + a_{qp}\} + \frac{t\delta}{2} \right\} \tag{4.4}$$

where

$$T_p = \begin{cases} \sum_{s=1}^{t} F_{ps} & \text{for } Y_p \in V^c \\ t\bar{F}_p(s_p) & \text{for } Y_p \in V \end{cases} .$$

In essence, for Y_q with $s_q > t$, $t\bar{F}_q(s_q)$ is substituted for $\sum_{s=1}^{t} F_{qs}$.

Step 4. *Stopping Rule*: If $|Q| = 1$, then stop and report the only survivor as the best; otherwise, for each $q \in Q$ and $Y_q \in V^c$, collect one additional response sample and set $t = t + 1$. If $t = R + 1$, terminate the procedure and select the solution in Q with the smallest sample mean as the best; otherwise, for each $q \in Q$ and $Y_q \in V$ with $s_q = t$, set $V^c = V^c \cup Y_q$ and $V = V \backslash Y_q$ and go to Step 3.

Figure 4.3. **Sequential Selection with Memory (adapted from [109])**

procedure of Figure 4.2 requires a single switch for each candidate that survives the screening step if a second stage is necessary. The SSM procedure requires a switch each time an additional sample is collected in Step 4 of Figure 4.3, which can potentially lead to a large number of switches if the number of candidates is large and if the candidates are nearly homogeneous in terms of mean response. The computational evaluation of Chapter 5 addresses the tradeoff between sampling costs and switching costs.

4.2 Use of Surrogate Models

To further address computational enhancements for MGPS-RS algorithms, an optional SEARCH step was implemented to exploit the flexibility of the pattern search framework with the goal of accelerating convergence to a near-optimal region of the design space. In particular, previously sampled points evaluated prior to and during the search are used to construct a surrogate function that approximates the true objective function. This function is then searched during the SEARCH step to nominate high quality trial points. If the surrogate is reasonably accurate and can be evaluated inexpensively relative to the cost of generating response samples, then the search may progress to good solutions with fewer cumulative samples than if no SEARCH step is used. Even if the initial surrogate is poor, convergence is still guaranteed and a savings may still be achieved (see [31]).

Paramount to the construction of surrogates is the selection of a family of plausible functions for use in approximating the true objective function. To avoid assuming a specific parametric representation of the underlying structure of f, an estimation technique is used from the nonparametric regression literature; the Nadaraya-Watson estimator [95, 151] is used to approximate the objective function at a point x according to (2.8). In this dissertation research, the commonly used multivariate Gaussian kernel function, originally proposed in univariate form by Parzen [105], is used. This results in the regression equation

$$\hat{f}(x) = \frac{\displaystyle\sum_{j=1}^{N} \bar{F}_j \exp\left(-\frac{D_j^2}{2h^2}\right)}{\displaystyle\sum_{j=1}^{N} \exp\left(-\frac{D_j^2}{2h^2}\right)} \tag{4.5}$$

where $D_j^2 = (x - x_j)^2$ represents the squared Euclidean distance from x to x_j and $h \geq 0$ is a smoothing parameter that determines the width of the kernel centered at each site x_j. For this reason, h is often called the *bandwidth*. The estimator $\hat{f}$ is referred to as the *surrogate* function. The regression function (4.5) has also been described in the context of *generalized regression neural networks* [135].

As discussed in Section 2.1.5, the estimator $\hat{f}$ can be thought of as the weighted average of all response means, $\bar{F}_j$, where the weight received by $\bar{F}_j$ depends on the distance between the corresponding x_j and the estimation point. The bandwidth h essentially determines the degree of nonlinearity in the surrogate function. As h increases, the curvature in $\hat{f}$ decreases such that, when h is very large, $\hat{f}$ is a constant that assumes the mean value of all $\bar{F}_j$; *i.e.*, $\frac{1}{N}\sum_{j=1}^{N} \bar{F}_j$. Smaller values of h allow more curvature in $\hat{f}$ but can cause outliers to have too great an effect on the estimate. If h is zero, $\hat{f}$ assumes the value of $\bar{F}_j$ for the corresponding x_j that is nearest the estimation point. The effect of the bandwidth value is illustrated in Figure 4.4, where the surrogate function (4.5) is fit to the following eight input/response pairs: (1, 180), (2, 189), (3, 170), (4, 188), (5, 207), (6, 212), (7, 196), and (8, 257). The figure shows that as the bandwidth increases, the surrogate function becomes less descriptive in terms of the curvature of the surface, eventually flattening to a horizontal line equalling the mean of the responses.

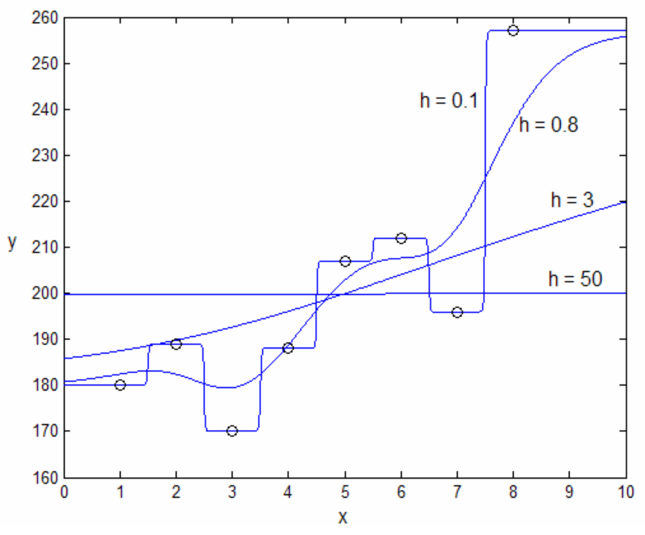


Figure 4.4. **Smoothing effect of various bandwidth settings for fitting a surface to eight design sites in one dimension.**

An advantage of the kernel regression approach is its simplicity. To evaluate a surrogate function at a design point, all that is required is storage of the pairs $(x_j, \bar{F}_j)$ and an "appropriate setting" for the lone bandwidth parameter. However, care must be taken in the practical consideration of selecting this setting. In this dissertation research, the well-known *leave-one-out cross-validation* method [55, p. 152] is used. In this method, the bandwidth is first set to a fixed value, and the estimator $\hat{f}$ is computed according to (4.5) at design site x_j, except that x_j is excluded from (*left out* of) the summand

$$\hat{f}_j(x_j, h) = \frac{\sum\limits_{\ell=1,\ell\neq j}^{N} \bar{F}_\ell \exp\left(-\frac{D_\ell^2}{2h^2}\right)}{\sum\limits_{\ell=1,\ell\neq j}^{N} \exp\left(-\frac{D_\ell^2}{2h^2}\right)}.$$

The squared error $(\hat{f}_j(x_j, h) - \bar{F}_j)^2$ is then recorded and summed over all sites x_j, $j = 1, \ldots N$. The resulting sum of squared errors (SSE),

$$SSE(h) = \sum_{j=1}^{N} (\hat{f}_j(x_j, h) - \bar{F}_j)^2,$$

is then used as a criterion for evaluating h. This procedure is repeated over a range of bandwidth values and the setting that delivers the smallest SSE is selected.

To build the original surrogate function prior to initiating the search, it is necessary to select design sites $x_1, \ldots, x_N$ via some appropriate experimental design technique. For this purpose, *latin hypercube sampling* (LHS) [88] is used. In LHS, a total of p equally-spaced values for each of the n^c continuous variables are used as components of the design site vectors. These values are randomly matched to form p design sites. If $N = p$, then each of the p values is represented exactly once in the set of design sites $x_1, \ldots, x_N$, and the design is said to be of *strength* one. Designs of strength two ($N = 2p$) are used in this dissertation research so that the design space is sampled more densely; the random matching operation is performed twice. Figure 4.5 illustrates latin hypercube samples of strengths one and two for a two-dimensional design space.

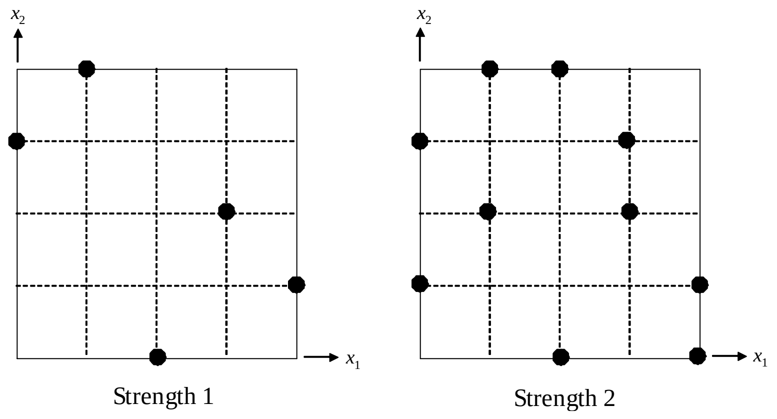


Figure 4.5. **Examples of Latin Hypercube Samples of Strengths 1 and 2 for $p = 5$.**

Once the surrogate function is built, it can be utilized in the pattern search framework as an inexpensive means to nominate trial points from the mesh M_k in the SEARCH step of the algorithm. After trial points are evaluated, they are then added as design sites that enhance the accuracy of the surrogate function. A straightforward approach is simply to minimize $\hat{f}$ on the mesh directly using any deterministic search routine. However, such a greedy approach may inhibit improvements in accuracy of the surrogate function because the trial points will cluster in a particular region of the design space.

Alternatively, the technique of Torczon and Trosset [145] is used to seek improvements in $\hat{f}$ while simultaneously seeking *space-filling* points that could improve the accuracy of the surrogate function. Torczon and Trosset [145] propose a biobjective function of the form,

$$m(x) = \hat{f}(x) - \lambda d(x) \tag{4.6}$$

where $d(x) = \min \|x - x_j\|_2$ is the distance from x to the nearest previously sampled design site and $\lambda \geq 0$ determines the relative weight placed on the space-filling objective. The function $m(x)$ is referred to as the *merit* function.

The surrogate function $\hat{f}$ is a smooth approximation to the unknown objective function with respect to the continuous variables. If one or more discrete variables change, then the true objective function may have an entirely different structure. Therefore, when using surrogates in a mixed-variable pattern search framework, it is necessary to maintain a surrogate function for each combination of discrete variables $i = 1, \ldots, i_{\max}$. Consequently, the number of initial design sites, surrogate function, merit function, bandwidth, and space-filling parameter are indexed as N_i, $\hat{f}_i$, m_i, h_i, and λ_i, respectively. If N_i design sites are selected for combination i, each requiring s samples of the response function, then a budget of $\sum_{i=1}^{i_{\max}} N_i \times s$ response samples is required during algorithm initialization.

The space-filling parameters λ_i need not remain constant throughout the search. In fact, as Torczon and Trosset [145] suggest, these parameters should tend to zero so that, after the surrogate is sufficiently accurate, the algorithm searches the surrogate directly. In the implementation, initial settings are used that are multiples of the maximum difference between mean responses of the initial design sites for each combination of discrete variable values. As the algorithm progresses, the parameters λ_i decay after each SEARCH step.

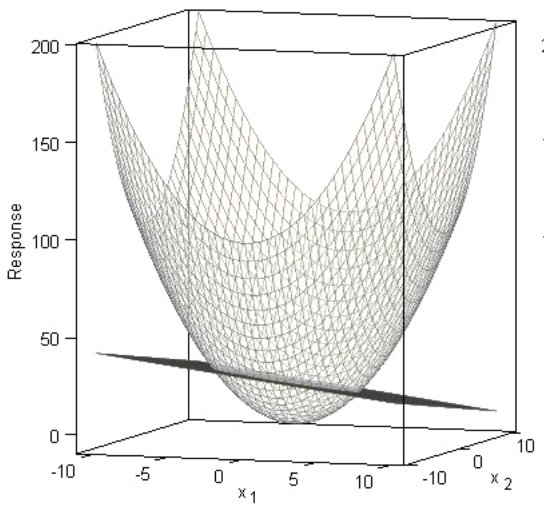
The MGPS-RS algorithm using surrogates is now illustrated on a very simple example that has two continuous variables and one discrete (binary) variable where each continuous variable is bounded on the range $[-10, 10]$. Consider the same additive noise response function (3.25) and true function (3.26) from Section 3.8. In this example, the functions f_1 and f_2 are linear and quadratic functions, respectively, that overlap in the continuous domain. Therefore, when the discrete variable x_3 is 0 (1), the function takes a linear (quadratic) form. The functions are defined as

$$\begin{aligned} f_1(x_1, x_2) &= 21 - x_1 - x_2, \text{ and} \\ f_2(x_1, x_2) &= x_1^2 + x_2^2. \end{aligned}$$

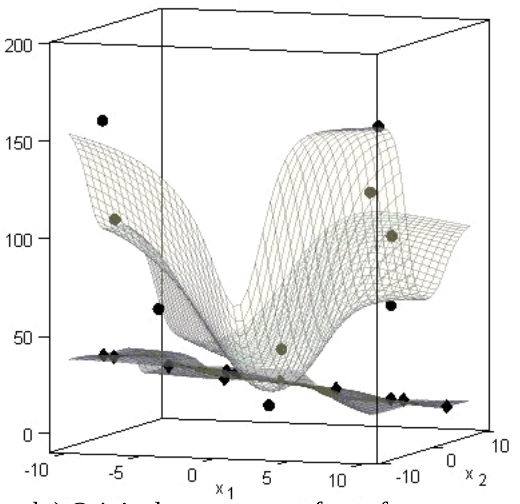
The optimum is located at $x^* = (0, 0, 1)$ with $f(x^*) = 0$ and the starting point was set to $x_0 = (-5, -5, 0)$ with $f(x_0) = 31$ so that the initial value of the discrete variable is suboptimal. The standard deviation of the noise term was set to $\sigma = 2$ throughout the design space.

A progression of the algorithm is illustrated Figure 4.6. For comparison purposes, the true function is shown in Figure 4.6a. In Figure 4.6b, the initial surrogate surfaces are shown for each of the two binary variable settings. A strength one LHS design with $p = 10$ was used to determine the initial design sites and five samples were taken at each design site. Therefore, a budget of $10 \times 5 \times 2 = 100$ response samples was necessary to build the original surrogates. Figure 4.6c shows the surrogate surfaces after nine iterations of the algorithm. By this point, 500 response samples had been generated and eight new design sites had been added to the surrogates. It can be seen how the space-filling parameter has forced the algorithm to evaluate points relatively far from the minimal point on either surrogate surface. Figure 4.6d shows the surfaces after 17 iterations and 2000 response samples. The search has now begun to cluster near the optimal point, as desired. Additionally, the form of the surface approximating the quadratic function appears to more accurately predict the true response.

Due to its simplicity, no notable improvements in the speed of convergence are attained for this example by using surrogates. However, the importance of improvements in surrogate accuracy is clearly illustrated. Note, for example, that in Figure 4.6b, the minimum on the



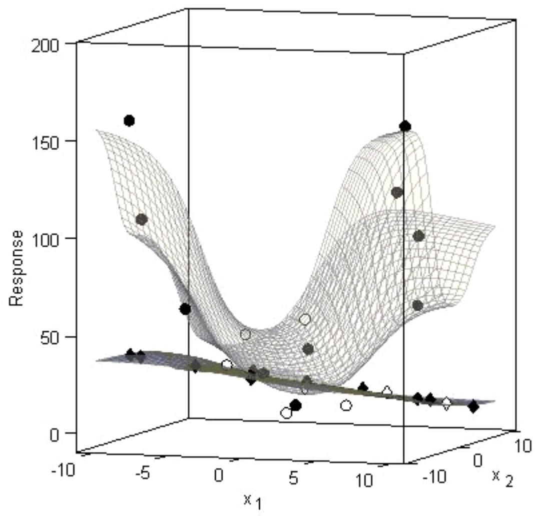
a.) True objective function



b.) Original surrogate surface after 100 response samples

◆ – sites sampled from linear function

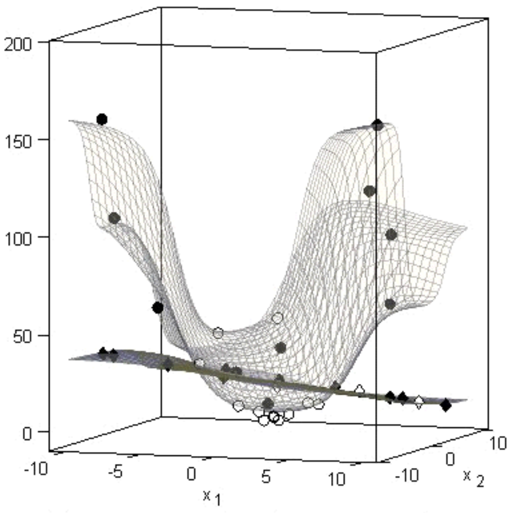
● – sites sampled from quadratic function



c.) Surrogate surface after 500 response samples (8 sites added)

◇– sites sampled from linear function (3)

○– sites sampled from quadratic function (5)



d.) Surrogate surface after 2000 response (15 sites added)

◇– sites sampled from linear function (3)

○– sites sampled from quadratic function (12)

Figure 4.6. **Demonstration of the surrogate building process during MGPS-RS algorithm execution.**

surrogate surfaces is actually located on the surface corresponding to the linear function f_1. By forcing the search to evaluate space-filling points, the accuracy of the surface corresponding to f_2 was eventually improved so that the surrogates correctly predicted a minimum on the surface of f_2. However, this behavior is not always guaranteed and presents some complications for problems with mixed variables. Fortunately, the algorithm provides the fail-safe EXTENDED POLL step that can ensure a search of alternative surfaces provided the extended poll trigger is set sufficiently large. For this reason, it may be beneficial to set this parameter large enough in early iterations to ensure that enough design points are sampled to enable accuracy improvements for each surrogate function $\hat{f}_i$.

4.3 Termination Criteria

An important consideration for algorithm implementation involves the decision of when to stop the algorithm. It is not uncommon for the termination decision to not be based on any particular strategy, allowing the algorithm to run until expending a fixed budget of iterations or response samples. For MGPS-RS algorithms, this can be disadvantageous because the search may reach a point where additional sampling leads to diminishing returns as the parameters α_r and δ_r get very small. Figure 3.6 demonstrated that marginal improvement can lead to an explosion in sampling requirements after a certain point in the search. This is further illustrated in Figure 4.7, where the MGPS-RS (without surrogates) sampling requirements per Rinott R&S procedure are plotted for problems of dimension two and ten. The sampling requirements are based on a direction set $D = [I, -I]$ consisting of positive and negative coordinate axes. In the figure, the R&S parameters are reduced geometrically ($\alpha_r = \alpha_0(0.95)^r$ and $\delta_r = \delta_0(0.95)^r$) from initial settings of $\alpha_0 = 0.8$ and $\delta_0 = 2.0$, the variance is assumed constant at $S^2 = 1$, and the number of first-stage samples is set to $s_0 = 5$. The number of response samples for each R&S procedure is computed as the product of

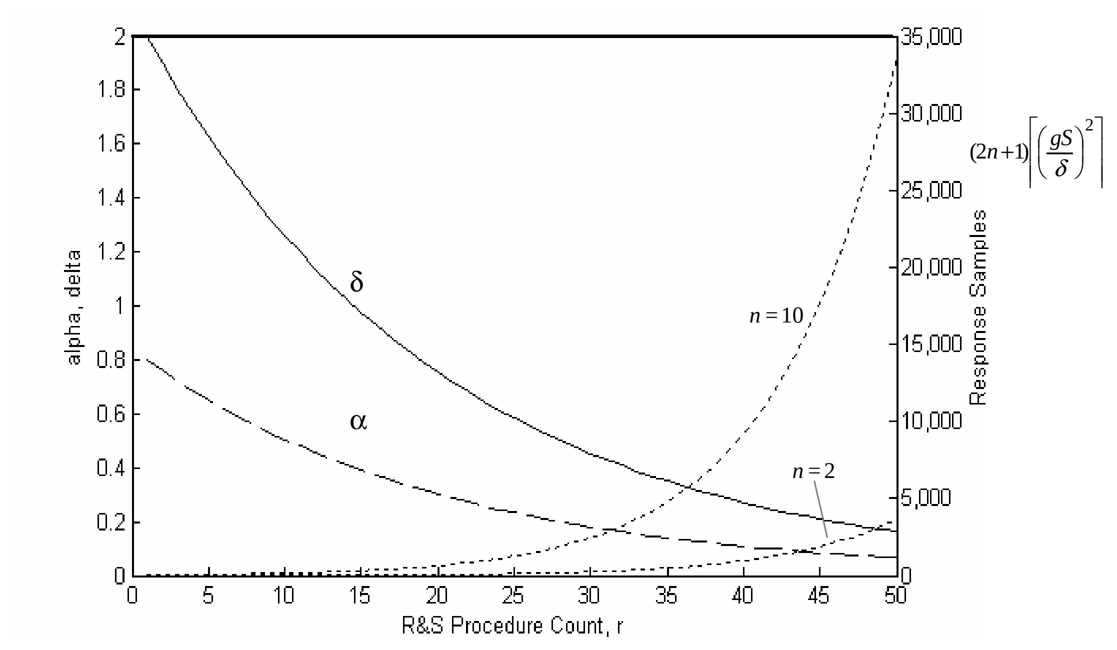


Figure 4.7. **Growth of response samples required per Rinott R&S procedure for a fixed response variance of $S^2 = 1$.**

the samples required per candidate using Rinott's second-stage formula $\left\lceil \left(\frac{gS}{\delta}\right)^2 \right\rceil$ and the number of candidates $n_C = 2n + 1$ (two for each dimension plus the incumbent).

Figure 4.7 demonstrates that, even for small problems, sampling requirements can become prohibitive (exceeding 30,000 for a single R&S procedure on a 10-dimensional problem after a modest number of iterations) if the parameters are reduced too aggressively or if the initial settings are too small. It would be advantageous if the algorithm could detect, based on some set of rules, a situation in which further progress would require a sampling effort that exceeds some threshold that reflects a budget restriction. For this purpose, Rinott's formula may be adapted for use as a heuristic tool to predict when sampling requirements become excessive according to a user-defined threshold. Using the formula, a per-iteration budgeting threshold B of response samples may be expressed as

$$B \approx g^2 n_C \left(\frac{S}{\delta_r}\right)^2$$

where $g = g(n_C, \alpha_r, s_0)$ increases with n_C and $1 - \alpha_r$. Observing that the size of the candidate set $n_C = n_C(n)$ is dependent on problem size, the budgeting threshold can be normalized with respect to problem size by setting it equal to a multiple of $g^2 n_C$, *i.e.,* $B = K(g^2 n_C)$ for some $K \in \mathbb{R}$. Therefore, a measure for estimating a point of minimal returns on sampling may be expressed in terms of the ratio between the response standard deviation and the indifference zone parameter as

$$\frac{S}{\delta_r} \geq \sqrt{K} \ . \tag{4.7}$$

The setting for K may be selected by the user based available sampling budget; larger values of K allow larger budgeting thresholds. As the algorithm progresses, response variance can be estimated by computing the sample variance S^2 of one of the candidates (*e.g.*, the incumbent) from an initial sampling stage and comparing its root to the current value of the indifference zone parameter. If the ratio exceeds the scalar $\sqrt{K}$, then one condition of termination may be considered satisfied.

Expression (4.7) has intuitive appeal because the difference between the two best candidates implied by δ_r can be expressed in terms of the standard deviation of the noise as $K^{-\frac{1}{2}}S$. However, this approach implicitly assumes that the value α_r has reached an appropriate level. That is, it is desirable for α_r to have reached a level to ensure sufficient error control (*e.g.*, $\alpha_r = 0.05$). To ensure that a desirable level of error control for iterate selection is reached by the end of the search, a secondary criterion is proposed that requires a sufficiently low value of α_r,

$$\alpha_r \leq \alpha_T \ , \tag{4.8}$$

where α_T is a threshold setting that defines the minimum desired probability of correct selection $1 - \alpha_T$ from among the candidates at termination. This measure provides a useful means to bound the probability of correct selection at an appropriate level by the end of the search, but does not prevent α_r from decreasing to such a small value that Rinott's constant g becomes very large, resulting in increased sampling independent of the ratio $\frac{S}{\delta_r}$. This can be seen in Figure 4.8, which shows the number of second-stage samples required using Rinott's procedure for two- and ten-dimensional problems with direction set $D = [I, -I]$, a fixed response standard deviation to indifference zone ratio of $\frac{S}{\delta} = 1$, and first-stage sampling size of $s_0 = 5$. The figure shows that sample size increases only moderately with $1 - \alpha$ until α gets very close to zero, when it increases dramatically.

Very small values in α_r have the same effect (increasing samples) using the SAS R&S procedure and a similar effect when using the SSM procedure. In the latter case, decreasing α_r causes increasing values of a_{qp} in (4.3) which then increases the tolerance in (4.4) used to screen candidates. The increased tolerance makes it more difficult to screen out the inferior candidates so that they are retained for additional samples when they would otherwise be eliminated from contention as the best design. An approach to algorithm design could allow the rate of decay of the α_r parameter to adapt during the search so that this parameter does not decay too aggressively and cause excessive sampling. However, adaptive parameter updates are an item for further study; in this research, the decay rate is determined *a priori* and its effect on sampling requirements and algorithm termination analyzed in the computational evaluation of Chapter 5.

A final criterion for termination invokes the traditional measure used in deterministic optimization via pattern search. In the original pattern search, Hooke and Jeeves [60] suggested terminating the search when the step size Δ_k reached a sufficiently small value,

$$\Delta_k \leq \Delta_T \ , \tag{4.9}$$

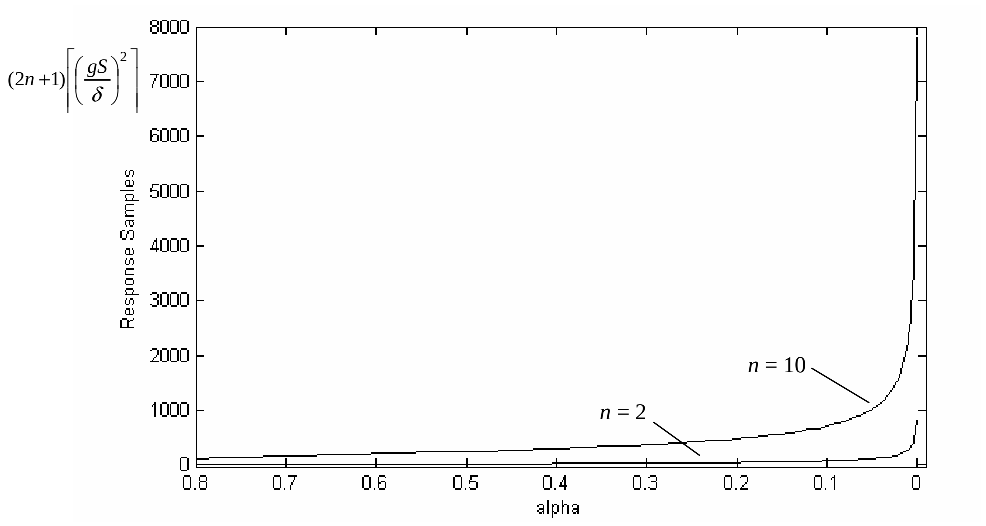


Figure 4.8. **Growth in response samples for Rinott's R&S procedure as α decreases for fixed ratio $\frac{S}{\delta} = 1$.**

for some small threshold setting Δ_T. This has long been used as a stopping criteria and has been justified analytically by Dolan *et al.* [40] by showing that Δ_k provides a bound on first-order stationarity measured in terms of the norm of the gradient $\|\nabla f(x_k)\|$.

In the present case, with noisy response functions, a combination of (4.7), (4.8), and (4.9) is proposed as criteria for terminating MGPS-RS algorithms. The condition (4.9) requires that enough unsuccessful iterations have occurred so that the poll set essentially converges to a set of points in near proximity to each other. However, condition (4.8) provides a safeguard to prevent a sequence of erroneous selections from causing the step size to decrease to a small value, prematurely terminating the algorithm if only the traditional criterion (4.9) were used. Finally, the intent of (4.7) is to provide a heuristic means to signal the onset of inflated sampling requirements caused by a high ratio of response noise to indifference zone parameter.

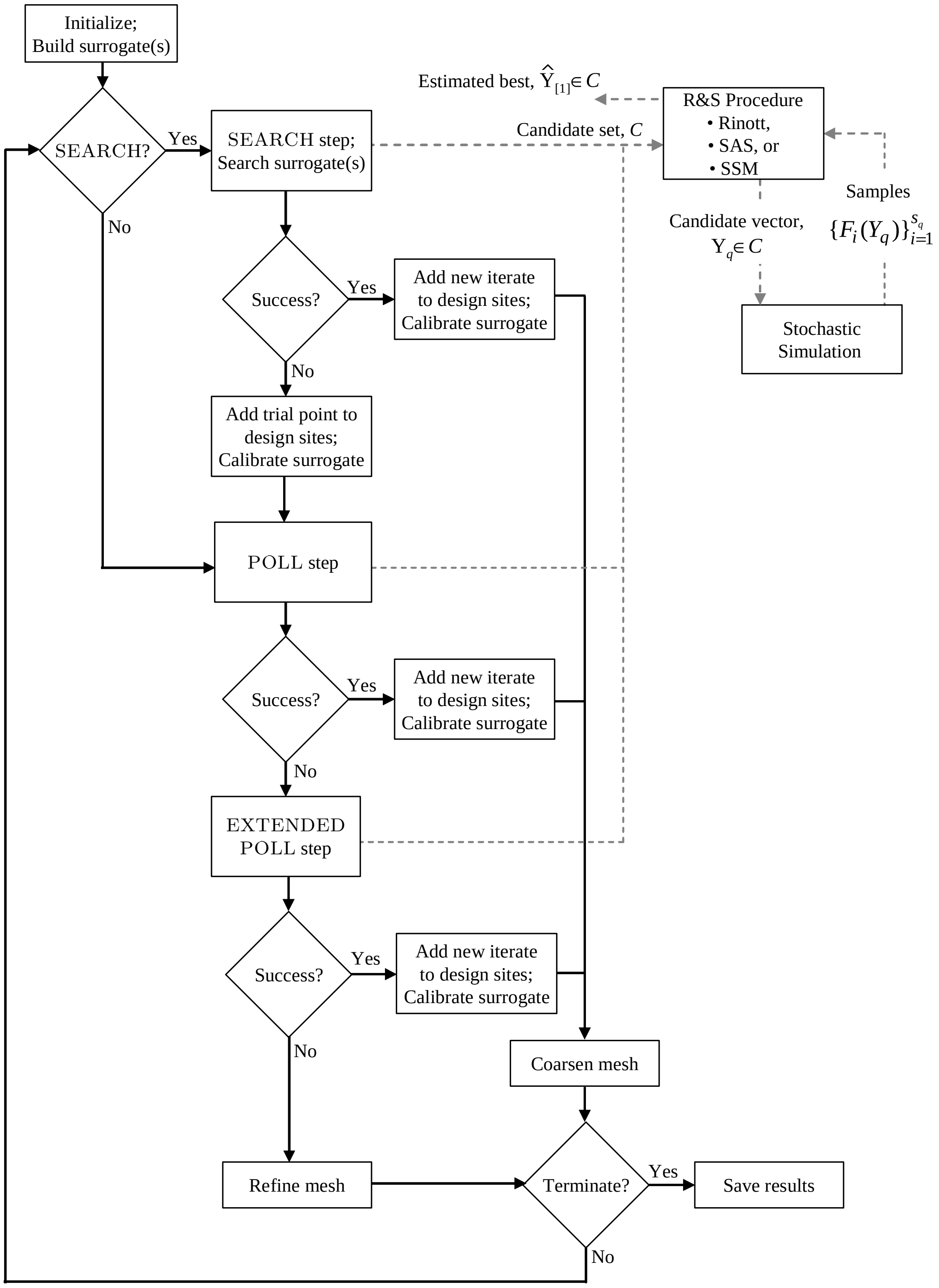
4.4 Algorithm Design

With the various implementation details defined, the overall design of the algorithm may now be described. The algorithm was coded in the MATLAB® programming language; a flow chart in Figure 4.9 shows the general sequence of steps taken by the algorithm code. The figure shows the interface between the algorithm and a stochastic simulation model via the R&S procedure. A more detailed mathematical description of the algorithm is shown in Figure 4.10, which is an update to Figure 3.4 to include the use of surrogates in the SEARCH step.

4.4.1 Building the Surrogate During Initialization

The first step in algorithm execution is the initialization step. When using surrogates, initialization requires building initial surrogate functions for each combination of design variables. Besides the number of design sites and samples per design site to select, there are other important surrogate-building considerations that must be taken into account. First, the boundaries of the region used for the initial latin hypercube sampling designs must be defined. If the variables are bounded, then the lower and upper limits of this region are simply set to the lower and upper bound vectors l and u, respectively. If some or all variables are unbounded, then the *range* for each variable must be decided on by the user and this becomes a parameter in the initialization step.

If there are linear constraints, then a second consideration involves what to do with design sites that are infeasible with respect to the linear constraints; that is, the feasible sampling region may be irregular (nonrectangular) due to the constraints. One approach is to simply discard infeasible design sites prior to sampling [32]. This is appropriate if the stochastic model is undefined for infeasible designs; however, it results in a loss of some design sites that can negatively impact accuracy of the surrogate. In this research, it is assumed that infeasible designs can be sampled and are retained in the set of design sites in order to improve surrogate accuracy, particularly in regions near the linear constraint boundaries.



Initialize; Build surrogate(s)
SEARCH?
Yes
No
SEARCH step; Search surrogate(s)
Estimated best, $\hat{Y}_{[1]} \in C$
Candidate set, C
R&S Procedure
• Rinott,
• SAS, or
• SSM
Samples
$\{F_i(Y_q)\}_{i=1}^{s_q}$
Candidate vector, $Y_q \in C$
Stochastic Simulation
Success?
Yes
No
Add new iterate to design sites; Calibrate surrogate
Add trial point to design sites; Calibrate surrogate
POLL step
Success?
Yes
No
Add new iterate to design sites; Calibrate surrogate
EXTENDED POLL step
Success?
Yes
No
Add new iterate to design sites; Calibrate surrogate
Coarsen mesh
Refine mesh
Terminate?
Yes
No
Save results

Figure 4.9. **Algorithm flow chart for MGPS-RS using surrogates.**

MGPS-RS Algorithm Using Surrogates

Initialization: For each combination of discrete variables $i = 1, \dots, i_{\max}$, do the following

- Select N_i design sites and set the number of samples s per design site.
- Collect the initial $N_i \times s$ samples and compute mean responses $\bar{F}_j^i$ for each design site $j = 1, \dots N_i$.
- Calibrate h_i using the leave-one-out cross-validation method, construct $\hat{f}_i$, and set space-filling parameter $\lambda_i \geq 0$.

Set the iteration counter k to 0. Set the R&S counter r to 0. Choose a feasible starting point, $X_0 \in \Theta$. Set $\Delta_0 > 0$, $\xi > 0$, $\alpha_0 \in (0, 1)$, and $\delta_0 > 0$.

Until termination criteria are satisfied, do the following:

1. SEARCH step: Find a candidate Y on the mesh $M_k(X_k)$ defined in (3.8) that minimizes $\cup_{i=1}^{i_{\max}} m_i$ where m_i is defined in (4.6). Use Procedure RS($\{Y, X_k\}$, α_r, δ_r) to return the estimated best solution $\hat{Y}_{[1]} \in \{Y, X_k\}$. Add Y as a design site and recalibrate the appropriate h_i, update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$ according to (3.12), and $k = k + 1$ and repeat Step 1. Otherwise, proceed to Step 2.

2. POLL step: Set extended poll trigger $\xi_k \geq \xi$. Use Procedure RS($P_k(X_k) \cup \mathcal{N}(X_k)$, α_r, δ_r) where $P_k(X_k)$ is defined in (3.9) to return the estimated best solution $\hat{Y}_{[1]} \in P_k(X_k) \cup \mathcal{N}(X_k)$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} \neq X_k$, the step is *successful*, add $\hat{Y}_{[1]}$ as a design site and recalibrate the appropriate h_i, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$ according to (3.12), and $k = k + 1$ and return to Step 1. Otherwise, proceed to Step 3.

3. EXTENDED POLL step: For each discrete neighbor $Y \in \mathcal{N}(X_k)$ that satisfies the extended poll trigger condition $\bar{F}(Y) < \bar{F}(X_k) + \xi_k$, set $j = 1$ and $Y_k^j = Y$ and do the following.

   a. Use Procedure RS($P_k(Y_k^j)$, α_r, δ_r) to return the estimated best solution $\hat{Y}_{[1]} \in P_k(Y_k^j)$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} \neq Y_k^j$, set $Y_k^{j+1} = \hat{Y}_{[1]}$ and $j = j + 1$ and repeat Step 3a. Otherwise, set $Z_k = Y_k^j$ and proceed to Step 3b.

   b. Use Procedure RS($X_k \cup Z_k$) to return the estimated best solution $\hat{Y}_{[1]} = X_k$ or $\hat{Y}_{[1]} = Z_k$. Update $\alpha_{r+1} < \alpha_r$, $\delta_{r+1} < \delta_r$, and $r = r + 1$. If $\hat{Y}_{[1]} = Z_k$, the step is *successful*, add $\hat{Y}_{[1]}$ as a design site and recalibrate the appropriate h_i, update $X_{k+1} = \hat{Y}_{[1]}$, $\Delta_{k+1} \geq \Delta_k$ according to (3.12), and $k = k + 1$ and return to Step 1. Otherwise, repeat Step 3 for another discrete neighbor that satisfies the extended poll trigger condition. If no such discrete neighbors remain, set $X_{k+1} = X_k$, $\Delta_{k+1} < \Delta_k$ according to (3.11), and $k = k + 1$ and return to Step 1.

Figure 4.10. **MGPS-RS Algorithm using Surrogates for Stochastic Optimization**

A third consideration involves defining the region of the design space within which the surrogate function can be trusted. This is relevant for unbounded problems in which a user-defined range must be established for the initial surrogate. Defining such a region is important because kernel regression methods are interpolatory since the regression surface approaches a constant hyperplane, with a value equal to the mean response of the nearest design site, outside the sampling region. In this research, the radius rad_{S} of the "searchable" region in the SEARCH step is approximated as one-half of the maximum Euclidean distance between any pair of the initial design sites. During the SEARCH step, the search is restricted to a ball centered at the starting point with radius rad_{S}.

A final consideration involves scaling the design sites so that they have approximately the same ranges when building and evaluating the surrogate function. Scaling is important because variable ranges may be quite different due to differing bounds but the bandwidth parameter prescribes the same width of the underlying Gaussian in each dimension. In the algorithm, scaling is accomplished by normalizing each variable of the design site vector; that is, subtracting the mean and dividing by the standard deviation. For design site $x_j = [x_j^1, x_j^2, \ldots, x_j^{n^c}]^T$, the normalized elements are represented as

$$\tilde{x}_j^\ell = \frac{x_j^\ell - \overline{x}^\ell}{\sigma_\ell}, \quad \ell = 1, \ldots, n^c$$

where $\overline{x}^\ell = \frac{1}{N}\sum_{j=1}^{N} x_j^\ell$ and $\sigma_\ell = \left(\frac{1}{N-1}\sum_{j=1}^{N}(x_j^\ell - \overline{x}^\ell)\right)^{1/2}$. Since the surrogate is built with respect to normalized design sites, then the trial points in the SEARCH step are also normalized before being evaluated with respect to the surrogate function.

The final steps conducted before initiating the search are, for each $i = 1, \ldots, i_{\max}$, to calibrate h_i using leave-one-out cross-validation, and assigning a value to the initial space-filling parameter λ_i. As mentioned in Section 4.2, initial settings of λ_i, $i = 1, \ldots, i_{\max}$, are used that are a multiple of the maximum difference between mean responses of the initial design sites for each combination of discrete variable values

$$\lambda_i = \theta_i \max_{j,\ell \in \{1,\ldots,N_i\}} \left| \bar{F}^i_j - \bar{F}^i_\ell \right|, \ i = 1, \ldots, i_{\max},$$

where the scalars θ_i become parameters that define λ_i, $i = 1, \ldots, i_{\max}$, during initialization. As the algorithm progresses, the parameters λ_i, $i = 1, \ldots, i_{\max}$, are halved after each SEARCH step.

4.4.2 Algorithm Search Steps and Termination

After initialization is complete, the search begins by seeking a mesh point that minimizes the merit function(s) (defined in (4.6)) in Step 1. In this research, pattern search is used for this purpose although, in general, any search procedure could be used, including a random draw from viable mesh points. Beginning from the incumbent, pattern search applied to the merit function is carried out in the continuous domain through a series of POLL steps using the current value of the step size parameter Δ_k and the direction set D^i to define the neighboring mesh points for discrete variable combination i. Once a local optimizer has been found for the current i, then the discrete neighbors at the optimal point are evaluated with respect to *their* merit function to check for further improvement. If no improvement is found among the discrete neighbors, then extended polling is conducted in the continuous neighborhood of *all* discrete neighbors to take advantage of the relative inexpense of evaluating the merit functions compared to response function sampling. The search of the merit function(s) terminates with the selection of a single trial point to be paired with the incumbent design in a candidate set that is passed to the R&S procedure. The SEARCH step culminates after the estimated best design is returned from the R&S procedure. Regardless of whether the trial point obtained in the SEARCH step successfully replaces the incumbent as the new iterate, its mean response is recorded so the point may be added as a new design site for the surrogate function in order to improve surrogate accuracy.

The POLL step and EXTENDED POLL step function as described in Section 3.6; however, if surrogates are used, an additional step is added after their termination. If either step results in a success, then the new iterate is added as a design site. The reasons why other trial points evaluated during the POLL step and EXTENDED POLL step are not added are twofold. First, these steps are more localized than the SEARCH step, so that the points evaluated will tend to cluster near the incumbent, giving artificially high weight in that region for surrogate evaluation. By contrast, the SEARCH step intentionally seeks points that help fill the experimental design space used to build the surrogate, so even trial points of unsuccessful SEARCH steps are worthy of adding as design sites for the purpose of surrogate accuracy enhancement. Secondly, if too many design sites are added, evaluating the surrogate function can become overly expensive, which defeats its purpose. This can be seen by reviewing Equation (4.5) and noting that the expression requires two summations of N terms where N is the number of design sites. In addition, the summation elements require the computation of the Euclidean distance between a trial point and a design site. A fine mesh can lead to many trial points evaluated during the SEARCH step, which can result in many evaluations of (4.5). Since many points may be evaluated during the POLL step and EXTENDED POLL step, it would be counterproductive to add all of them as design sites; therefore, only new iterates after successful steps are added. Any time a new design site is added, the current bandwidth may not provide the minimum sum of squared error over the set of augmented design sites. For this reason, the algorithm calls the calibration routine to recalibrate the appropriate bandwidth parameter and improve the accuracy of the surrogate.

It should also be mentioned that at the beginning of the POLL step, the direction set D_k is updated to ensure conforming directions are included when near the constraint boundaries. The algorithm used for computing conforming directions was adapted from Abramson [1, p. 49], which is equivalent to the original algorithm of Lewis and Torczon [81]. For completeness, the algorithm listing is shown in Figure 4.11. This algorithm is valid in the absence of degenerate constraints.

The use of surrogates to augment the search is a valuable enhancement to the algorithm. However, the portion that ensures its rigor in a stochastic setting is the R&S procedure. In the algorithm, the R&S procedure manages the interface with the stochastic model by passing the design variable vector of each candidate to the model and prescribing the number of response samples necessary to meet the correct selection probability guarantee. Depending on the specific procedure used – Rinott's, SAS, or SSM – the procedure may also need to manage the overhead necessary to repeatedly switch between candidate designs to gather the required samples.

```
Set $\epsilon_k \geq \epsilon > 0$. Assume the current iterate satisfies $l \leq AX_k \leq u$.

While $\epsilon_k \geq \epsilon$, do the following:
  1. Let $I_l(X_k, \epsilon_k) = \{i : AX_k - l \leq \epsilon_k\}$
  2. Let $I_u(X_k, \epsilon_k) = \{i : u - AX_k \leq \epsilon_k\}$
  3. Let $V$ denote the matrix whose columns are formed by all members of the set
     $\{-a_i : i \in I_l(X_k, \epsilon_k)\} \cup \{a_i : i \in I_u(X_k, \epsilon_k)\}$, where $a_i^T$ denotes the $i$th row of $A$.
  4. If $V$ does not have full column rank, then reduce $\epsilon_k$ just until $|I_l(X_k, \epsilon_k)| + |I_u(X_k, \epsilon_k)|$
     is decreased, and return to Step 1.
Set $B = V(V^TV)^{-1}$ and $N = I - V(V^TV)^{-1}V^T$.

Set $D_k = [N, -N, B, -B]$.
```

Figure 4.11. **Algorithm for Generating Conforming Directions (adapted from [1] and [81]).**

Once an iteration has been deemed successful or unsuccessful and the appropriate mesh updates are completed, the final decision to be made before initiating another iteration is whether or not to terminate the algorithm. For this purpose, criteria (4.7) – (4.9) may be assessed for compliance against user defined thresholds. A word of caution is repeated here that was introduced in Section 4.3. If the α_r parameter decays too fast, then very small values may force large per-iteration samples before the step size Δ_k decreases to a sufficiently small value. In the absence of adaptive decay rates for α_r or δ_r, it may be prudent to make criterion (4.9) optional so that, when the per-iteration sampling requirements become enormous prior to $\Delta_k \leq \Delta_T$, the algorithm may be stopped.

4.4.3 Algorithm Parameters

The algorithm design may be concluded by summarizing the various parameter settings required. For reasons discussed in Section 4.3, perhaps the most critical parameters with regard to performance are the R&S parameters δ_r and α_r. These parameters have the most influence over the number of samples required for each R&S procedure executed by the algorithm. In practice, it is desirable to avoid excessive sampling in regions of the search space far from optimality. An advantage of the MGPS-RS algorithms is that through manipulation of these parameters, the sampling requirements can be increased gradually as the algorithm progresses, so that excessive sampling effort is not wasted at early iterations. In this study, each parameter is reduced geometrically with r,

$$\delta_r = \delta_0(\rho_\delta)^r \quad \text{and} \quad \alpha_r = \alpha_0(\rho_\alpha)^r.$$

The initial values δ_0 and α_0 are set very loose so that, in the early iterations, no samples are taken beyond the initial s_0 required for each candidate in all three procedures used.

As the algorithm progresses, error control of iterate selection increases as the search moves toward the region of optimality.

The adjustable algorithm parameters are summarized in Table 4.1. The parameters may be grouped into three general categories:

- mesh defining parameters D, Δ_0, τ, m_k^-, m_k^+, ξ, and ξ_k;
- R&S parameters δ_0, α_0, ρ_δ, ρ_α, and s_0; and
- surrogate defining parameters p, *strength*, *range*, θ_i, and $[h_{\text{low}}, h_{\text{high}}]$.

Some additional parameters are implicitly defined by parameters in the table. For example, the number of design sites N_i to build the initial surrogate are defined as the prod-

Table 4.1. **Summary of MGPS-RS parameters.**

Parameter	Description
D	Direction set used for mesh definition, must be positive spanning; common choices are $D = [I, -I]$ and $D = [I, -e]$ where e is a vector of ones
Δ_0	Initial step size, must satisfy $\Delta_0 > 0$
τ	Mesh update parameter, constant for all k, must satisfy $\tau > 1$ and $\tau \in \mathbb{Q}$
m_k^-	Mesh refinement parameter, must satisfy $-\infty < m_k^- \leq -1$ and $m_k^- \in \mathbb{Z}$, can vary by iteration
m_k^+	Mesh coarsening parameter, must satisfy $0 \leq m_k^+ < +\infty$ and $m_k^+ \in \mathbb{Z}$, can vary by iteration
ξ, ξ_k	Extended poll trigger and lower bound on trigger, must satisfy $\xi_k \geq \xi > 0$, ξ_k can vary by iteration
δ_0	Initial indifference zone setting, must satisfy $\delta_0 > 0$
α_0	Initial significance level setting, must satisfy $0 < \alpha_0 < 1$
ρ_δ	Indifference zone decay parameter, must satisfy $0 < \rho_\delta < 1$
ρ_α	Significance level decay parameter, must satisfy $0 < \rho_\alpha < 1$
s_0	Number of response samples for R&S procedure initial stage, must satisfy $s_0 \geq 2$
p	Number of intervals for each continuous dimension in LHS design, must satisfy $1 \leq p < \infty$, $p \in \mathbb{Z}$
strength	Strength of LHS design, small, positive integer (*e.g.* 1 or 2)
range	Limits on sampling region for LHS design
θ_i	Factor to determine initial setting of space-filling parameter λ_i, must satisfy $\theta_i \geq 0$
$[h_{\text{low}}, h_{\text{high}}]$	Allowable range on bandwidth parameter h_i, must satisfy $0 < h_{\text{low}} \leq h_{\text{low}} < \infty$

uct of the *strength* of the latin hypercube sampling design and the number of intervals p selected for each continuous dimension. Another example is the initial space-filling parameter λ_i, which is defined as θ_i times the maximum difference in mean response between any two design sites in the initial LHS design for discrete variable combination i. The values for h_i, $i = 1, \ldots, i_{\max}$ are determined by the leave-one-out cross-validation method in the calibration routine, but must be within bounds $[h_{\text{low}}, h_{\text{high}}]$. The bounds $[h_{\text{low}}, h_{\text{high}}]$ are necessary to prevent overfitting or underfitting the surrogate function to the design sites. Values of h_i that are too small will cause overfitting in the sense that the surface of the surrogate function will pass through or very close to the response value at each design site with very sharp drop-offs in between sites; this results from too much weight given to the nearest design site. Values of h_i that are too large will cause underfitting in the sense that the surface of the surrogate function passes further way from the response value at each design site with a more gradual slope between sites; this results from too much weight given to designs sites far from the nearest site. An illustration of underfitting and overfitting was shown in Figure 4.4.

4.5 Summary

Building on the framework presented in Chapter 3, this chapter added detail regarding the various algorithm implementations with specific regard to computational concerns. The use of modern R&S techniques to improve sampling efficiency and use of surrogate functions to accelerate convergence were highlighted as general approaches to enhance computational performance of the basic algorithm. Additionally, a strategy for algorithm termination was proposed which seeks to detect the onset of excessive sampling requirements and avoid additional sampling if only marginal improvement is expected. In the next chapter, the impact of the various implementations is assessed in a comprehensive computational evaluation.

Chapter 5 - Computational Evaluation

A computational evaluation was conducted to assess the performance of the various implementation strategies presented in the preceding chapter. This evaluation consisted of a series of experiments that applied the different algorithm variants to a suite of standardized test problems. To complement the evaluation, four additional algorithms from the literature were implemented in order to compare their performance to the MGPS-RS algorithms. Following an overview of the test scenario in Section 5.1, the competing algorithms, and their implementation details under this computational study, are presented in Section 5.2. The test problems used for the evaluation, which consist of twenty-two continuous-variable and four mixed-variable problems, are described in Section 5.3. In Section 5.4, the design of experiments is presented which defines the performance measures, outlines the statistical model to be evaluated, and lists the parameter settings for each of the algorithms. The numerical results are analyzed in Section 5.5. Special attention is given to a study of the effects of the various MGPS-RS implementation alternatives, the comparison of MGPS-RS to its competitors used in this evaluation, and the effectiveness of the termination criteria proposed in the preceding chapter.

5.1 Test Scenario

To test the algorithm implementations of Chapter 4, the generic response function

$$F(x) = f(x) + N(0, \sigma^2(f(x))$$

is used, where $N(0, \sigma^2(f(x))$ is a normally distributed, mean-zero noise term added to an underlying true objective function. Standard test functions were drawn from the literature to compose $f(x)$, some of which are constrained by variable bounds, linear constraints, or both. A total of 26 test problems were defined, four of which contain mixed variables. The test problems are described in greater detail in Section 5.3 and Appendix A.

To compare two different random noise scenarios, the standard deviation of the noise term $\sigma(f(x))$ is either proportional or inversely proportional to f, but bounded on the range $(0.1, 10)$:

$$\begin{aligned} \sigma_1(f(x)) &= \min\left(10, \sqrt{f(x) - f(x^*) + 1}\right), \quad \text{or} \\ \sigma_2(f(x)) &= \max\left(0.1, \frac{1}{\sqrt{f(x) - f(x^*) + 1}}\right), \end{aligned}$$

where $f(x^*)$ is the known optimal solution. These test cases are referred to as noise cases 1 and 2, respectively. At optimality, $\sigma_1 = \sigma_2 = 1$ but diverge to values $\sigma_1 = 10$ and $\sigma_2 = 0.1$ for trial points away from optimality. The noise cases were selected to provide both a *high* noise case (case 1) and a *low* noise case (case 2) and also to demonstrate that the MGPS-RS algorithms allow for the inclusion of modern R&S procedures that do not require known and/or constant variance of response samples across different designs.

5.2 Competing Algorithms

Testing was performed for each of the following six MGPS-RS variants:

- MGPS with Rinott's procedure and no surrogates (MGPS-RIN),
- MGPS with Screen-and-Select procedure and no surrogates (MGPS-SAS),
- MGPS with Sequential Selection with Memory procedure and no surrogates (MGPS-SSM),
- Surrogate assisted MGPS with Rinott's procedure (S-MGPS-RIN),
- Surrogate assisted MGPS with Screen-and-Select procedure (S-MGPS-SAS), and
- Surrogate assisted MGPS with Sequential Selection with Memory procedure (S-MGPS-SSM).

For comparison to other methods, four additional algorithms were included in computational experiments:

- Finite-Difference Stochastic Approximation (FDSA),
- Simultaneous Perturbation Stochastic Approximation (SPSA),
- Nelder-Mead simplex search (NM), and
- a Random Search (RNDS) algorithm.

Each of the ten methods was coded in the MATLAB® programming language. The MGPS-RS implementations are described in Section 4.4. Code for FDSA and SPSA was obtained from the web site associated with the textbook of Spall [134] and modified as necessary. Code for NM was adapted from the MATLAB® function `fminsearch`. Code for RNDS was written by the author. The details of the algorithms are provided in the following paragraphs. Due to algorithm limitations, FDSA, SPSA and NM were not applied to the test problems with mixed variables. In addition, NM was not applied to the constrained test problems. The RNDS algorithm was adapted for all test problem types.

The FDSA and SPSA methods are based on the algorithm in Figure 2.1 using (2.2) to estimate the gradient for FDSA and (2.3) for SPSA. As recommended by Spall [134, p.113], the step sizes for FDSA and SPSA are updated according to

$$a_k = \frac{a}{(k+1+A_{\text{SA}})^{\alpha_{\text{SA}}}} \tag{5.1}$$

with constant scalar parameters $a > 0$, $\alpha_{\text{SA}} > 0$, and $A_{\text{SA}} \geq 0$. The parameter $A_{\text{SA}} \geq 0$ is designed to provide stability in the early iterations when a is large enough to ensure nonnegligible step sizes after many iterations.

The perturbation distance parameter c_k, used by (2.2) and (2.3) to specify interval width in the gradient approximation, is updated as per Spall [134, p. 163]

$$c_k = \frac{c}{(k+1)^{\gamma_{\text{SA}}}} \tag{5.2}$$

with constant scalar parameters $c > 0$ and $\gamma_{\text{SA}} > 0$. Spall [134, p. 190] suggests setting c equal to the approximate standard deviation of the response noise. For nearly all test problems, the setting $c = 1$ is used. In the remaining ones, a smaller value was needed to prevent the formulae in (2.2) and (2.3) from evaluating the objective at infeasible points, where it is not defined. These test cases are elaborated on further in Section 5.4.

The forms of (5.1) and (5.2) ensure that the iterates of the SA algorithms are asymptotically normally distributed about the optimal solution [134, p. 162], which provides a means to determine rates of convergence. It can be shown that the optimal *asymptotic* rate of convergence is achieved for settings of $\alpha_{\text{SA}} = 1$ and $\gamma_{\text{SA}} = 1/6$. However, for better finite-sample algorithm performance, Spall [134, p. 190] recommends using values $\alpha_{\text{SA}} = 0.602$ and $\gamma_{\text{SA}} = 0.101$, which are the lowest possible settings that satisfy the theoretical conditions necessary to retain normally distributed iterates. These settings are used throughout the computational testing. For SPSA, the elements of the perturbation direction vector d_k are drawn randomly from a Bernoulli ± 1 distribution with probability $\frac{1}{2}$ for each outcome. Each point in the differencing formula is averaged over $s_0 = 5$ response samples for both FDSA and SPSA.

As suggested by Spall [134, p. 165], A_{SA} is selected to be approximately 10% of the total number of iterations. Therefore, for $RS_{\max}$ total response samples allowed for each algorithm run, A_{SA} is determined as

$$\begin{aligned} A_{\text{SA}} &= 0.1\left(\frac{RS_{\max}}{2n^c s_0}\right) \quad \text{for FDSA, and} \\ A_{\text{SA}} &= 0.1\left(\frac{RS_{\max}}{2s_0}\right) \quad \text{for SPSA.} \end{aligned}$$

Given A_{SA} and an initial desired step size a_0, the constant a is selected semiautomatically after an initial NS number of response samples obtained at the starting point according to the methods suggested in [134, p. 165, 190]. For SPSA, a is determined as

$$a = \frac{a_0 \left(A_{\text{SA}} + 1\right)^{\alpha_{\text{SA}}}}{\hat{g}_0(X_0)},$$

where $\hat{g}_0(X_0)$ is the mean of the estimated gradient vector elements averaged over the NS responses. For FDSA, a is determined as $a = \min\{a_{\text{temp},1}, a_{\text{temp},2}, \ldots, a_{\text{temp},n^c}\}$, where

$$a_{\text{temp},i} = \frac{a_0 \left(A_{\text{SA}} + 1\right)^{\alpha_{\text{SA}}}}{\hat{g}_{0,i}(X_0)}$$

and $\hat{g}_{0,i}(X_0)$ is the estimated gradient vector element for coordinate i averaged over the NS responses. The value $NS = 200$ is used for SPSA and $NS = \max(200, 2n^c s_0)$ for FDSA. A larger number of samples is used for FDSA (for test problems exceeding 20 variables) because FDSA uses more samples per gradient estimate as n^c increases, whereas SPSA always uses $2s_0$ samples. Given the preceding discussion for determining parameter settings, the only parameter left that requires tuning for both FDSA and SPSA prior to algorithm execution is the initial desired step size a_0 (except for c in just three of the test problems).

For problems with variable bounds, elements of infeasible iterates $X_{k,i}$, $i = 1, \ldots . n^c$ produced by FDSA and SPSA are set to l_i (u_i) for lower (upper) bound violations. Handling the linear constraints is a bit more complicated, however. In these cases, a mapping to the feasible region was attempted through a sequence of corrective moves in the negative direction of the outward pointing normal vector to the maximum violated constraint. This simple technique is illustrated for two dimensions in Figure 5.1, which shows two cases in which the two constraints f_1 and f_2 are violated. On the left side of the figure, a single move is required to map iterate X_k to feasible point X_k'. On the right side of the figure, an infinite number of moves are actually needed to reach the intersection of f_1 and f_2 in the limit. To avoid computational deficiencies resulting from a very large number of attempted moves,

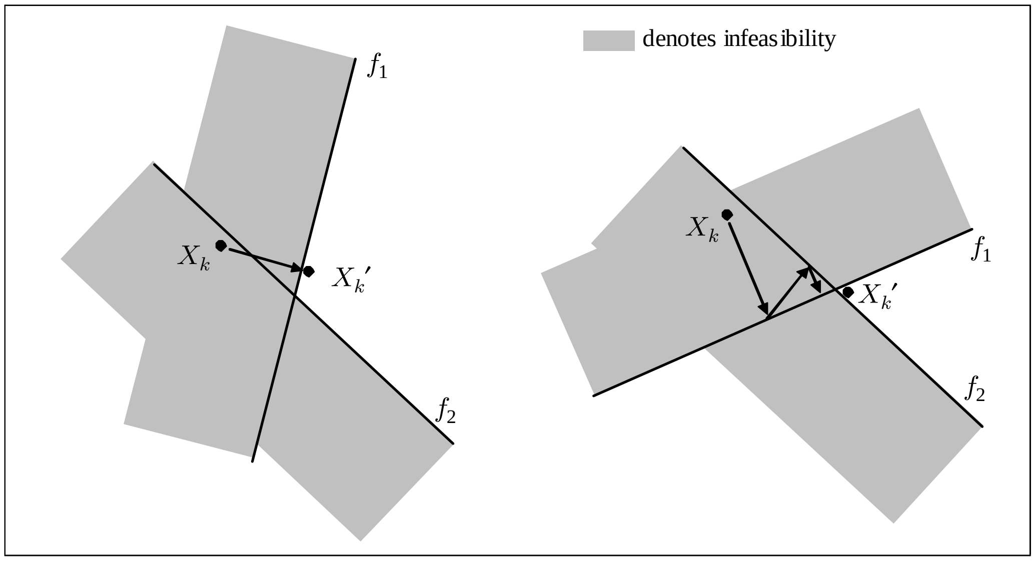


Figure 5.1. **Illustration of Corrective Move Method for Infeasible Iterates Used in SA Algorithms.**

the number of moves is limited to 50 per iteration in algorithm implementation. Thus, in the presence of multiple linear constraints, the corrected iterates are not guaranteed to be feasible but will be closer to the feasible region than the pre-correction iterates.

The Nelder-Mead algorithm is based on the algorithm listing in Figure 2.3. As suggested by Barton and Ivey [23], the shrink parameter was adjusted to $\kappa = 0.9$ (from 0.5), while all others were set according to the standard choices [152]: reflection parameter $\eta = 1$, expansion parameter $\gamma_{\mathrm{NM}} = 2$, and contraction parameter $\beta = \frac{1}{2}$. In addition, the best point is resampled after a shrink. Any time a point in the simplex is sampled for the first time or resampled, the objective function value is evaluated as the mean response over $s_0 = 5$ samples. The initial simplex is constructed using the starting point plus n^c points a distance of Δ_{NM} units in the direction of the coordinate axes from the starting point. The parameter Δ_{NM} was tuned for each of the test problems to try and find an initial simplex size that allowed the search to achieve the best results. Details of parameter tuning procedures are provided in Section 5.4.2.

The Random Search algorithm is based on the general algorithm of Figure 2.2, adapted to a mixed-variable domain. The specific algorithm used in the computational evaluation is shown in Figure 5.2. The neighborhood structure for the continuous domain is based on the algorithm in [134, p. 45], which is a simplification of an algorithm in [131]. In Step 1, the continuous portion of a trial point is generated by perturbing the incumbent $(X_k^c)' = X_k^c + b_k + d_k$ where b_k is a bias vector and d_k is a normally distributed random perturbation vector with mean zero vector and covariance $(\rho_k^R)^2 I$. The parameter ρ_k^R allows the standard deviation of the perturbation terms, which is set equal for all dimensions, to be adjusted by iteration. In the algorithm of Figure 5.2, the value is reduced by a factor of 0.99 after each iteration so that, after many iterations, the magnitude of the perturbation d_k gets smaller as the optimum is approached. The initial standard deviation parameter ρ_0^R was tuned for each of the test problems to try and find an initial setting that allowed the search to achieve the best results. Details of parameter tuning procedures are provided in Section 5.4.2. The b_k vector slants the search of a candidate in the direction of previous success. The discrete portion of a trial point is randomly assigned by selecting a combination of discrete variable settings i uniformly from all the possible settings and using the values for X^d that correspond to i.

In Step 2 of the algorithm, the mean responses of the incumbent and trial points are averaged over k samples. Therefore, precision in estimating the objective function increases with k so that, early in the search, exploration of the design space is encouraged due to a greater likelihood of accepting trial points even if the *true* objective value does not improve upon the incumbent. As the number of iteration grows, it becomes increasingly difficult to replace good iterates because of increased precision in the estimates. If the first trial does not successfully replace the incumbent, then a second is tried by reversing the direction of

Random Search Algorithm

Initialization: Set $b_0 = 0$ and initial setting for $\rho_0^R > 0$. Choose a feasible starting point $X_0 \in \Theta$.

Set the iteration counter k to 0.

1. Generate independent random vector $d_k \sim NORMAL(0, \rho_k^R I)$ and independent random integer $i \sim UNIF(1, i_{\max})$. Construct trial point $X_k' = \left((X_k^c)', (X_k^d)'\right)$ according to following:
   - $(X_k^c)' = X_k^c + b_k + d_k$, and
   - $(X_k^d)' = (X^d)_i$ corresponding to the ith combination of discrete variables.

2. Obtain k response samples $\{F_s(X_k)\}_{s=1}^{k}$ for the incumbent design point and calculate mean response $\bar{F}(X_k) = k^{-1} \sum_{s=1}^{k} F_s(X_k)$.
   - If X_k' is infeasible, set $\bar{F}(X_k') = \infty$; otherwise,obtain k response samples $\{F_s(X_k')\}_{s=1}^{k}$ for the trial design point and calculate mean response $\bar{F}(X_k') = k^{-1} \sum_{s=1}^{k} F_s(X_k')$. If $\bar{F}(X_k') < \bar{F}(X_k)$, set $X_{k+1} = X_k'$ and $b_{k+1} = 0.2b_k + 0.4d_k$ and go to Step 3.
   - Set $(X_k^c)' = X_k^c + b_k - d_k$ and keep current $(X_k^d)'$. If X_k' is infeasible, set $\bar{F}(X_k') = \infty$; otherwise, obtain k response samples $\{F_s(X_k')\}_{s=1}^{k}$ for the trial design point and calculate mean response $\bar{F}(X_k') = k^{-1} \sum_{s=1}^{k} F_s(X_k')$. If $\bar{F}(X_k') < \bar{F}(X_k)$, set $X_{k+1} = X_k'$ and $b_{k+1} = b_k - 0.4d_k$ and go to Step 3.
   - Set $X_{k+1} = X_k$ and $b_{k+1} = 0.5b_k$

3. If the stopping criteria is satisfied, then stop and return X_{k+1} as the estimate of the optimal solution. Otherwise, update $\rho_{k+1}^R = 0.99\rho_k^R$ and $k = k + 1$ and return to Step 1.

Figure 5.2. **Random Search Algorithm Used in Computational Evaluation**

the perturbation vector d_k. If neither trial replaces the incumbent, the bias vector is halved before returning for another iteration. Note that response samples for infeasible points are avoided by setting the response mean value to a very large number such that it cannot be accepted as the new iterate.

5.3 Test Problems

The test problems are drawn from standardized problem sets in [58] and [122]. In total, twenty-six test problems are used for the computational evaluation – twenty-two of them with continuous variables only and four of them with mixed variables. The mixed-variable problems are constructed similarly to the example of Section 3.8 in that $f(x)$ takes a specific functional form depending on the settings of the discrete variables. The following subsections summarize the test problems.

5.3.1 Continuous-Variable Problems

The continuous-variable test problems are drawn from the published collections in [58] and [122], the latter being a supplement to the former. Taken together, these books present a total of 307 problems as "an extensive set of nonlinear programming problems that were used by other authors in the past to develop, test or compare optimization algorithms" [122, p. iii]. The collection consists of unconstrained and constrained problems that range in dimension from two to 100. A nice feature of the publications is that they provide a classification scheme to help characterize the structure of each problem. The objective function (OBJ) is classified as one of the following categories:

- constant (C),
- linear (L),
- quadratic (Q),
- sum of squares (S),
- generalized polynomial (P), or
- general nonlinear (G).

The constraint information is classified as one of the following categories:

- unconstrained (U),
- upper and/or lower bounds only (B),
- linear constraint functions (L),
- quadratic constraint functions (Q),
- generalized polynomial constraint functions (P), or
- general nonlinear constraint functions (G).

In this dissertation research, an objective of the test problem set is that it represents a cross section of the relevant objective function and constraint category combinations. Since the MGPS-RS algorithms are applicable in an unconstrained setting or under bound and linear constraints, the constraint categories available for testing are U, B, and L. In addition, very few of the problems with linear objective functions have only linear constraints (none that are unconstrained or bounded only); therefore, objective function categories are restricted to Q, S, P, and G. Finally, it is deemed important to stratify algorithm performance based on problem size. Therefore, an additional category considered in this research was established based on problem dimension. The categories are defined as:

- small (S) – 2 to 9 variables,
- medium (M) – 10 to 29 variables, or
- large (L) – 30 to 100 variables.

Note that the "large" category here is not necessarily representative of large *practical* problems, which may include thousands of design variables.

With four objective function categories, three constraint type categories, and three problem size categories, the total number of problem type combinations numbers $4 \times 3 \times 3 = 36$. Of the thirty-six combinations, twenty-two are satisfied by at least one of the published problems, which led to the conclusion to select twenty-two continuous-variable test problems.

A summary of the continuous-variable test problems is displayed in Table 5.1. The problem number shown is the number assigned in [58] or [122]. The table lists the objective function type, problem dimension (and size category), and constraint information to include the number of bounds and/or linear constraints. A more detailed description of the test problems, included in Appendix A, provides the objective and constraint equations, the starting solution, and the optimal solution. It should be noted that the published starting point was used for each problem except for problem 392. In this case, the published starting

Table 5.1. **Continuous-Variable Test Problem Properties.**

Problem Number	OBJ	CON	DIM	Number of Bounds	Number of Linear Constraints
3	Q	B	2 (S)	1	0
4	Q	B	2 (S)	2	0
5	G	B	2 (S)	4	0
25	S	B	3 (S)	6	0
36	P	L	3 (S)	6	1
105	G	L	8 (S)	16	1
110	G	B	10 (M)	20	0
118	Q	L	15 (M)	30	29
224	Q	L	2 (S)	4	4
244	S	U	3 (S)	0	0
256	P	U	4 (S)	0	0
275	Q	U	4 (S)	0	0
281	G	U	10 (M)	0	0
287	P	U	20 (M)	0	0
288	S	U	20 (M)	0	0
289	G	U	30 (L)	0	0
297	S	U	30 (L)	0	0
300	Q	U	20 (M)	0	0
301	Q	U	50 (L)	0	0
305	P	U	100 (L)	0	0
314	G	U	2 (S)	0	0
392	Q	L	30 (L)	45	30

point is infeasible. Since MGPS-RS algorithms search the interior of the feasible region, the starting point was modified from the published version to make it feasible.

5.3.2 Mixed-Variable Problems

The mixed-variable problems are constructed by assigning a specific functional form to the objective function over the continuous domain $f(x^c)$ based on the settings of the discrete variables x^d. To simplify test problem construction, one discrete variable is used with a varying number of settings $i_{\max}$. For testing, two settings of $i_{\max}$ (2 and 3) are used as well as two settings for the dimension n^c (4 and 20) to comprise a total of four test problem combinations. The following variably dimensioned test functions were selected so that their form could be adjusted to the dimension n^c:

$$\begin{aligned}
f_1(x^c) &= \sum_{\ell=1}^{n^c-1} [(x^c_{\ell+1} - x^c_\ell)^2 + (1 - x^c_\ell)^2], \\
f_2(x^c) &= 5 + 5(x^c)^T Q x^c, \qquad \text{where } Q(i,j) = \frac{1}{i+j-1}, \text{ and} \\
f_3(x^c) &= 2 + 20n^c - 5\sum_{\ell=1}^{n^c} x^c_\ell.
\end{aligned}$$

The first two functions are variations on functions presented in [122]. Function f_1 is a version of the well-known Rosenbrock banana function, several versions of which are presented in [122] (problems 206–210, 294–299). Function f_2 is a quadratic function using the Hilbert matrix to generate the coefficients on the terms; three versions of this function are presented in [122] (problems 274-276). The scalar multipliers of these functions were adjusted so that their surfaces do not deviate from each other too much in the feasible design space (*e.g.* the scalar "5" was introduced as a multiplier to the $(x^c)^T Q x^c$ term). In addition, a constant term was added to function f_1 so that its minimum value does not coincide with that of f_1. Both functions f_1 and f_2 are used to define the objective function when $i_{\max} = 2$; the linear function f_3 is added when the value of $i_{\max}$ is increased from 2 to 3.

The four mixed-variable test problems are summarized in Table 5.2. For each of the problems, all continuous variables are bound on the range $[-4, 4]$. The lone discrete variable x^d has either two (MVP1 and MVP3) or three (MVP2 and MVP4) settings, which determine the form taken by the objective function. For all test problems, the optimal solution corresponds to $x^* = (x^c_1, x^c_2, \ldots, x^c_{n^c}, x^d)^* = (0, 0, \ldots 0, 1)$, $f(x^*) = 0$. The continuous portion of the starting point was selected as the standard starting point for problems 274-276 in [122], *i.e.*, $x^c_\ell = -4/\ell$, $\ell = 1, \ldots, n^c$. The discrete portion was selected as $x^d = 2$ or $x^d = 3$. The test problems are shown graphically in Figure 5.3 for a two-dimensional case.

Table 5.2. **Mixed-variable Test Problems.**

Test Problem	$i_{\max}$	Objective function	DIM(n^c)	Bounds on x^c	Starting Point
MVP1	2	$f(x^c) = \begin{cases} f_1(x^c), & \text{if } x^d = 1 \\ f_2(x^c), & \text{if } x^d = 2 \end{cases}$	4	$[-4, 4]$	$x^c_\ell = 4/\ell$, $\ell = 1, \ldots, 4$ $x^d = 2$
MVP2	3	$f(x^c) = \begin{cases} f_1(x^c), & \text{if } x^d = 1 \\ f_2(x^c), & \text{if } x^d = 2 \\ f_3(x^c), & \text{if } x^d = 3 \end{cases}$	4	$[-4, 4]$	$x^c_\ell = 4/\ell$, $\ell = 1, \ldots, 4$ $x^d = 3$
MVP3	2	$f(x^c) = \begin{cases} f_1(x^c), & \text{if } x^d = 1 \\ f_2(x^c), & \text{if } x^d = 2 \end{cases}$	20	$[-4, 4]$	$x^c_\ell = 4/\ell$, $\ell = 1, \ldots, 20$ $x^d = 2$
MVP4	3	$f(x^c) = \begin{cases} f_1(x^c), & \text{if } x^d = 1 \\ f_2(x^c), & \text{if } x^d = 2 \\ f_3(x^c), & \text{if } x^d = 3 \end{cases}$	20	$[-4, 4]$	$x^c_\ell = 4/\ell$, $\ell = 1, \ldots, 20$ $x^d = 3$

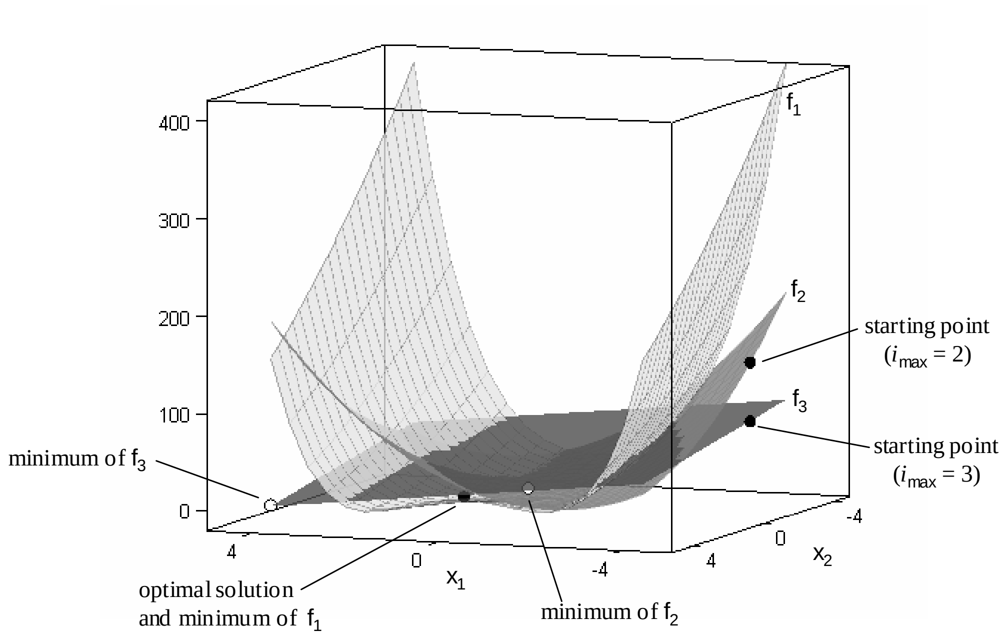


Figure 5.3. **Mixed-variable Test Problem Illustration for $n^c = 2$.**

5.4 Experimental Design

The computational experiments constitute a full factorial design, where for each valid algorithm, test problem, and noise case combination, thirty independent replications were executed. In each experiment the algorithm was allowed to run until $RS_{\max} = 100,000$

response samples were obtained (small and medium problems) or until $RS_{\max} = 500,000$ response samples were obtained (large problems and problems MVP3 and MVP4).

5.4.1 Performance Measures and Statistical Model

Three performance measures are defined to evaluate the numerical results. Since the experiments were executed across a number of different PC-based platforms, the performance measures do not include computer processing time, which is not always a consistent indicator of algorithm quality even on a standard platform. The following performance measures are used, where Q and P are used in the comparison of all algorithms and SW is used to compare the MGPS-RS variants to each other. In the performance measure definitions, $x^*(f^*)$ refers to the optimal design vector (objective function value), $x_0(f_0)$ to the starting design vector (objective function value), and $x(f)$ to the final design vector (objective function value) after the search.

- Solution quality,
$$Q = \frac{f - f^*}{f_0 - f^*};$$
- Proximity to the true optimal:
  - Continuous-variable problems,
$$P = \frac{\|x - x^*\|}{\|x_0 - x^*\|};$$
  - Mixed-variable problems,
$$P = \frac{\|x^c - x^{c*}\| + \min(1, |x^d - 1|)}{\|x_0^c - x^{c*}\| + 1}; \text{ and}$$
- Number of cumulative switches, SW, due to the R&S procedure.

The measures Q and P are scaled by dividing by the absolute difference between the starting and optimal values, thereby providing a dimensionless quantity that allows consistency in comparisons across test problems. In the mixed-variable case, the measure P is defined as it is so that if the discrete variable has not reached the optimal setting at termination, the numerator is penalized by one unit regardless of which suboptimal setting it may have. This is in keeping with the concept that categorical variables have no ordering so that a setting of "3" should not be perceived as further than "2" from an optimal value of "1".

For each experiment involving a MGPS-RS variant, the following statistical model is postulated for performance measure Q,

$$\begin{aligned} Q_{ijk\ell} &= \beta_0 + \beta_R W_{R_i} + \beta_S W_{S_j} + \beta_N W_{N_k} + \beta_{RS} W_{R_i} W_{S_j} \\ &\quad + \beta_{RN} W_{R_i} W_{N_k} + \beta_{SN} W_{S_j} W_{N_k} + \varepsilon_{ijk\ell} \end{aligned} \tag{5.3}$$

where $1 \le \ell \le 30$ is the replication index, $1 \le k \le 2$ is the noise case index, $1 \le j \le 2$ is the surrogate index, $1 \le i \le 3$ is the R&S index, and "coded" independent variables W_{R_i}, W_{S_j}, and W_{N_k} represent experimental design factors for R&S, use of surrogates, and noise case, respectively. The design factors are defined as

$$\begin{aligned} W_{R_i} &= \begin{cases} -1, \text{ for } i = 1 \text{ (RIN)}, \\ 0, \text{ for } i = 2 \text{ (SAS)}, \\ +1, \text{ for } i = 3 \text{ (SSM)}; \end{cases} \\ W_{S_j} &= \begin{cases} -1, \text{ for } j = 1 \text{ (no surrogates)}, \\ +1, \text{ for } j = 2 \text{ (surrogates)}; \end{cases} \quad \text{and} \\ W_{N_k} &= \begin{cases} -1, \text{ for } k = 1 \text{ (noise case 1)}, \\ +1, \text{ for } k = 2 \text{ (noise case 2)}. \end{cases} \end{aligned}$$

A similar model is postulated for performance measure P.

5.4.2 Selection of Parameter Settings

To avoid excessive parameter tuning, a subset of parameter settings for all algorithms are kept constant throughout the experiments. For the SA algorithms and NM, the recommended settings discussed in Section 5.2 are used when appropriate. Parameter tuning cannot be completely avoided, however, and those parameters that are adjusted for each continuous-variable test problem are identified with an entry of "tune" in Table 5.3, which

Table 5.3. **Parameter Settings for All Algorithms – Continuous-variable Problems.**

	Parameter	Setting	Parameter	Setting
MGPS-RS	D_0	$\left[\ I,\ \ -I\ \right]$	δ_0	100
	τ	2	α_0	0.8
	Δ_0	tune	ρ_δ	0.95
	m_k^-	-1	ρ_α	0.95
	m_k^+	0	s_0	5
	p	10	θ	10
	strength	2	*range*	tune
	$[h_{\text{low}}, h_{\text{high}}]$	$[.1, 3]$		
FDSA	α_{SA}	0.602	c	$1^{(\text{Note 1})}$
	γ_{SA}	0.101	s_0	5
	NS	depends on n^c	A_{SA}	depends on n^c and $RS_{\max}$
	a_0	tune		
SPSA	α_{SA}	0.602	c	$1^{(\text{Note 2})}$
	γ_{SA}	0.101	s_0	5
	NS	200	A_{SA}	depends on $RS_{\max}$
	a_0	tune		
NM	κ	0.9	γ_{NM}	2
	η	1	β	0.5
	Δ_{NM}	tune	s_0	5
RNDS	ρ_0^R	tune		

Note 1: $c = .01$ (problem 105) and $c = .5$ (problem 110)
Note 2: $c = .25$ (problem 25), $c = .01$ (problem 105) and $c = .5$ (problem 110)

summarizes the parameter settings for all algorithms. Note that alternative settings for the FDSA and SPSA parameter c are used for some test problems. Each of the test problems requiring this change involves evaluating a logarithm in the objective function, and the parameter modification was necessary to ensure that the gradient estimator does not try to take the logarithm of a negative number resulting from a larger setting for c.

Parameter tuning for the "tunable" parameters was carried out informally by running each algorithm for a few thousand response samples and observing the output. Care was taken to ensure that the SA algorithms did not diverge or seem unstable, as they are prone to do if a_0 is set too large, and that the MGPS-RS variants, RNDS and NM seemed to

Table 5.4. **Summary of "Tunable" Parameter Settings for all Algorithms – Continuous-variable Problems.**

Problem	MGPS-RS		FDSA			SPSA		NM	RNDS
Number	Δ_0	*range*	a_0	A_{SA}	NS	a_0	A_{SA}	Δ_{NM}	ρ_0^R
3	.5	10	.1	500	200	.1	1000	NA	1.0
4	.25	2.5	.1	500	200	.02	1000	NA	.5
5	.5	bounds	.1	500	200	.02	1000	NA	1.0
25	2.0	bounds	.02	333	200	.05	1000	NA	2.5
36	1.0	bounds	.05	333	200	.01	1000	NA	2.0
105	.25	bounds	.002	125	200	.001	1000	NA	1.0
110	.1	bounds	.1	100	200	.03	1000	NA	.25
118	4	bounds	2	67	200	.25	1000	NA	4
224	.5	bounds	.5	500	200	.5	1000	NA	.5
244	2	5	.1	833	200	.1	1000	8	1.5
256	1	5	1	250	200	.05	1000	8	.5
275	1	2.5	.25	250	200	.25	1000	2	1
281	.5	2.5	.1	100	200	.02	1000	8	.5
287	1	4	2.5	50	200	1	1000	2	.5
288	1	4	1	50	200	.5	1000	1	.5
289	.1	2	.001	167	300	.01	5000	2	.1
297	2	2.5	2	167	300	.3	5000	10	1
300	.1	1	.05	50	200	.05	1000	500	.5
301	2	10	.01	100	500	.008	5000	500	.25
305	2	5	.3	50	1000	.05	5000	2	2
314	.25	1	.3	500	200	.05	1000	10	.1
392	10	bounds	10	167	300	1	5000	NA	10

achieve reasonable progress. The final settings for these parameters for each algorithm and test problem are displayed in Table 5.4.

For the mixed-variable test problems, the parameter settings used for the MGPS-RS variants and RNDS in all experiments are displayed in Table 5.5. Two items regarding these settings are worthy of mention. First, the extended poll trigger ξ_k is set to a large value during the early iterations and then reset to a smaller value after the algorithm conducts two EXTENDED POLL steps. This ensures extended polling is conducted so that samples are generated at alternate settings of x^d if the original surrogates do not, due to surrogate inaccuracies, correctly predict improving designs during the SEARCH step at the alternate x^d settings (which would avoid such sampling). Secondly, the R&S decay parameters ρ_δ and

Table 5.5. **Parameter Settings for MGPS-RS and RNDS Algorithms – Mixed-variable Problems.**

	Parameter	Setting	Parameter	Setting
MGPS-RS	D_0	$[\ I, \quad -I\ ]$	δ_0	100
	τ	2	α_0	0.8
	Δ_0	.5	ρ_δ, ρ_α (MVP1,2)	0.95
	m_k^-	-1	ρ_δ, ρ_α (MVP3,4)	0.99
	m_k^+	0	s_0	5
	ξ_k (MVP1,2)	200 (10)$^{\text{Note}}$	θ_i	10
	ξ_k (MVP3,4)	2000 (20)$^{\text{Note}}$	*range*	bounds
	strength	2	p	10
	$[h_{\text{low}}, h_{\text{high}}]$	$[.1, 3]$		
RNDS	ρ_0^R	2		

Note: The notation $X(x)$ indicates that ξ_k is initially set to X but reset to x after two EXTENDED POLL steps are executed.

ρ_α are set to a higher value (0.99) for the larger problems (MVP3 and MVP4). This ensures that the parameters α_r and δ_r decay at a slower rate with the hope that the algorithm will not exhaust an excessive portion of the response sampling budget prematurely through a sequence of R&S procedures conducted during the EXTENDED POLL steps.

5.5 Results and Analysis

The computational experiments were run on a total of 26 different PC workstations. The computational platforms ranged in processing speed from 2.00 GHz to 3.00 GHz, in random access memory from 256 MB to 2.0 GB, and operated under the Windows 2000 or Windows XP operating systems. Three of the platforms had dual processors.

The following sections summarize the quantitative analysis of the results. Additional test result data are presented in various forms in Appendix B. For example, to give a visual perspective of algorithm progression, a series of graphs are displayed that plot performance measure Q, averaged over 30 replications, versus the number of response samples obtained for each algorithm, noise case, and test problem.

5.5.1 Analysis of MGPS-RS Variant Implementations

To evaluate the effect of the various implementation options on performance measures Q and P within MGPS-RS, a formal analysis of variance (ANOVA) was performed on the statistical model (5.3) using the JMP$^{\text{TM}}$ 5.1 statistical software [121]. To assess the validity of the model, the estimated studentized residuals were examined using normal probability plots and the Shapiro-Wilk test for normality [116, 125]. In most cases, the data $Q_{ijk\ell}$ and $P_{ijk\ell}$ required a transformation to approximately satisfy the normality and constant variance assumptions required by the ANOVA procedure. The commonly used transformations suggested in Montgomery [93, p. 84] were used, which include the square root, natural logarithm, reciprocal square root, and reciprocal transformations. Even after transformation, the Shapiro-Wilk test frequently rejected the null hypothesis that the residuals were normally distributed at the .05 significance level, perhaps because the transformed residual distributions remained slightly skewed and there was a large sample size (360 residuals – one from each combination of 6 algorithms, 30 replications, and 2 noise cases). For a large number of sample points, the cumulative deviation from normality, used in computing the test statistic, can be more dramatic than for smaller samples, causing the test to fail. Furthermore, it proved difficult to attain approximately constant variance of the residuals, which was assessed graphically by plotting the residuals versus fitted values. The ANOVA procedure is typically robust to moderate departures from normality in the residuals (see Montgomery [93, p. 77]), and since the sample size is large and balanced (equal samples for each of the factors W_R, W_S, and W_N), then the normality and constant variance assumptions may be approximately satisfied. Included in Appendix B is a listing of transformations used, the results of the Shapiro-Wilk test, normal probability plots and residual versus fitted values plots for each test problem.

The ANOVA procedure was used to determine the significance of the effects β_R, β_S, β_N, β_{RS}, β_{RN}, and β_{SN} in model (5.3). To investigate which R&S procedures led to better results in the event that β_R was significant, the ANOVA was followed up by a multiple comparison test at the .05 significance level on the transformed data to compare means $\bar{T}(Q_1)$, $\bar{T}(Q_2)$, and $\bar{T}(Q_3)$, where $\bar{T}(Q_i)$ is defined as

$$\bar{T}(Q_i) = \frac{1}{120}\sum_{j=1}^{2}\sum_{k=1}^{2}\sum_{\ell=1}^{30} T(Q_{ijk\ell}), \qquad i = 1, 2, \text{ or } 3, \tag{5.4}$$

and $T(\cdot)$ denotes the transformation (the same test was used for $\bar{T}(P_i)$, $i = 1, 2, 3$, where $\bar{T}(P_i)$ is as defined in (5.4)). The multiple comparison test employed the Tukey honestly significant difference (HSD) procedure (see Sheskin [126, p. 534]) that makes all possible pairwise comparisons and tests for significant differences among the means, grouping them accordingly. Mean performance measures assigned to the same group are not statistically different from each other under this test.

As a safety precaution in the event of violated model assumptions, a battery of non-parametric statistical procedures was also used to test for differences among the factor populations. In particular, the following methods were used:

- the Wilcoxon rank-sum test [126, p. 289] for two factor levels (W_S and W_N) or the Kruskal-Wallis one-way ANOVA [126, p. 597] for three factor levels (W_R),
- the two-sample median test [71, p. 304] for two factor levels (W_S and W_N) or the Brown-Mood k-sample test [71, p. 315] for three factor levels (W_R), and
- the van der Waerden test [126, p. 611] for any number of factor levels (W_S, W_N, and W_R).

The Wilcoxon rank-sum procedure tests the hypothesis that the median of two sample populations are different. The Kruskal-Wallis procedure can be considered an extension of the Wilcoxon procedure that tests whether at least two of k sample populations have different median values. The two-sample median procedure tests whether two populations have the same cumulative density function (c.d.f.) by categorizing each sample from both populations according to whether or not it is above or below the composite median value and counting the instances of each category. The Brown-Mood procedure extends this to k sample populations and tests whether at least two of the populations have a different c.d.f. The van der Waerden procedure also tests whether at least two of k sample populations come from different distributions. This procedures organizes the data into a set of rank-orders, then transforms the rank-orders into a set of normal scores (z scores) from a standard normal distribution. If the average of the normal scores of all populations are not equal at a prescribed significance level, then the null hypothesis that the populations derive from the same c.d.f. is rejected.

The results of the significance tests on the main effects β_R, β_S, and β_N are displayed in Tables 5.6 (continuous-variable problems) and 5.7 (mixed-variable problems). In the figures, the absence of any entry indicates that the effect corresponding to effect of that column tested as insignificant for that test problem. For example, the effect β_R was insignificant for performance measure P on test problem 3 so the choice of R&S procedure had no effect toward proximity to the true optimal solution at termination. For effect β_S, an entry "+" indicates that employing surrogates had a positive effect (indicating improvement) toward the performance measure where an entry "–" indicates a negative effect. Similarly, for effect β_N, an entry "+" indicates that going from noise case 1 to 2 had a positive effect toward the performance measure where an entry "–" indicates a negative effect. For effect β_R, the entry indicates the results of the Tukey HSD multiple comparison test where the groups are listed in descending order of performance measure quality in terms of the transformed data. For example, in test problem 3, using Rinott's procedure resulted in a better and statistically different mean $\bar{T}(Q_1)$ than using SSM ($\bar{T}(Q_3)$), but $\bar{T}(Q_2)$ (SAS) was not

Table 5.6. **Significance Tests of Main Effects for Performance Measures Q and P – Continuous-variable Test Problems .**

Test Problem	Q			P		
	β_R	β_S	β_N	β_R	β_S	β_N
3	1–SAS, RIN 2–SAS, SSM	+	+		−	
4		+	+		+	+
5		+	+			+
25		+	−		+	+
36	1–RIN, SSM 2–RIN, SAS (∗)	+	+		+ (∗)	+
105		+	+		+	+
110			+			+
118		+	+		+	+
224			+			+
244		+	+		+	+ (∗)
256	1–SAS, SSM 2–SAS, RIN	+	+		+ (∗)	+
275			+			
281	1–SAS, SSM 2–SAS, RIN	+	+	1–SSM 2–SAS, RIN		+
287	1–RIN 2–SAS, SSM (∗)	+	+	1–SAS, RIN 2–SAS, SSM (∗)	+	+
288	1–SSM 2–RIN 3–SAS	−	+	1–SSM 2–RIN 3–SAS	− (∗)	+
289	1–SSM, SAS 2–SSM, RIN (∗)	−		1–SSM, SAS 2–SSM, RIN (∗)	−	
297	1–SSM, RIN 2–SAS	−	+	1–SSM 2–RIN 3–SAS		+
300	1–SSM, SAS 2–RIN (∗)	+	+		+	+
301	1–SSM 2–RIN 3–SAS	+ (∗)	+	1–SSM, RIN 2–SAS		+
305	1–SSM 2–RIN 3–SAS	+ (∗)	+	1–SSM 2–SAS, RIN	+ (∗)	+
314	1–RIN, SAS 2–RIN, SSM	+ (∗)	+		(∗)	+
392		−	+	(∗)	−	+

Table 5.7. **Significance Tests of Main Effects for Performance Measures Q and P – Mixed-variable Test Problems.**

Test Problem	Q			P		
	β_R	β_S	β_N	β_R	β_S	β_N
MVP1	1–SSM 2–SAS, RIN		+	1–SSM, RIN 2–SAS, RIN (∗)		+
MVP2	1–SSM 2–SAS, RIN	(∗)	+	1–SSM 2–SAS, RIN		+
MVP3	1–SSM 2–SAS, RIN		+	1–SSM 2–SAS, RIN		+
MVP4	1–SSM 2–SAS, RIN	+ (∗)	+	1–SSM 2–SAS, RIN (∗)		+

statistically different from $\bar{T}(Q_1)$ or $\bar{T}(Q_3)$. Finally, an entry (or nonentry) accompanied by a symbol "(∗)" indicates that at least two of the three nonparametric procedures were in disagreement with the ANOVA and/or multiple comparison results. For example, in test problem 36, all three nonparametric tests actually failed to be significant; the Kruskal-Wallis test did not detect a difference between any of the medians of the observation populations $Q_{1jk\ell}$, $Q_{2jk\ell}$, or $Q_{3jk\ell}$ ($1 \leq j \leq 2$, $1 \leq k \leq 2$, $1 \leq \ell \leq 30$) nor did the Brown-Mood or van der Waerden test detect a difference in the distributions from which those data were drawn. More detailed results of the nonparametric tests are included in Appendix B.

The results indicate a strong agreement between the ANOVA/multiple comparison tests and the nonparametric tests. Of the 180 possible tests for significance of main effects, only 20 are contradicted by at least two nonparametric tests, which provides some validation that the ANOVA results may be justified. Most of the disagreements occur with respect to the effect of the R&S procedure and the use of surrogates, typically refuting the possible effects predicted by the ANOVA procedure.

Not surprisingly, the results show that better results are almost always achieved in noise case 2 (low noise) than in noise case 1 (high noise), having a positive effect on Q and P in 24 and 23, respectively, of the 26 problems using the ANOVA results. This demonstrates the adverse effects that high response noise can have on solution quality over a fixed budget of response samples.

An encouraging finding is that, for 16 of 26 problems (better than 60%), the use of surrogates had a positive effect on solution quality Q at termination using the ANOVA results. It is interesting to note that in four of the problems, the use of surrogates actually had a negative effect on Q. This occurred for some of the larger problems (one with $n^c = 20$ and three with $n^c = 30$) and may be due to the fact that the number of design sites used to build the original surrogate was not increased as the size of the problem increased. As a result, the surrogate functions for the larger problems may have suffered from larger relative inaccuracies that caused the algorithm to search unpromising regions of the design space in vain. The effect of surrogate use on P is similar to that of Q but not as pronounced; in ten instances the use of surrogates had a positive effect (each corresponding to one of the 16 positive effect instances for Q) and in four instances had a negative effect (three of them corresponding to the four instances for Q). For problem 3, the use of surrogates actually positively affected Q but negatively affected P.

Similar observations can be made regarding the use of the various R&S procedures. For performance measure Q, sixteen of the 26 problems showed significant effects for parameter β_R. The multiple comparison tests revealed that procedure SSM was in the lead group (delivering the smallest mean $\bar{T}(Q_i)$) on thirteen occasions, where SAS and RIN were in the lead group on six and five occasions, respectively. In the case of measure P, effect β_R was significant eleven times and SSM, SAS, and RIN were in the lead group on ten, two, and three occasions, respectively. These results seem to indicate an advantage of using the SSM procedure over the range of test problems, but it is interesting to note that RIN is not always dominated by either SSM or SAS when β_R is significant. Perhaps the conditions under which the more modern procedures SSM and SAS perform well (heterogeneity of true objective function values among the candidate set) do not occur as frequently as expected, at least for the test problems and noise structure considered in this research. On the other hand, it should be mentioned that the choice of R&S procedure shows an effect with greater frequency for the large problems than for the small and medium problems. Furthermore, Rinott's procedure is in the lead group for the large problems on only one occasion for Q (shared with SSM) and never for P. This suggests that choice of the R&S procedure becomes more critical as problem dimension grows.

Tables 5.6 and 5.7 do not make reference to the interaction terms β_{RS}, β_{RN}, and β_{SN} in model (5.3). For the most part, these terms failed to be significant. Term β_{SN} was significant in thirteen problems for measure Q and nine problems for measure P. Each of the terms β_{RS} and β_{RN} was significant in either five or six problems for both measures. Furthermore, no systematic trend exists for the cases that tested significant. For example, the term β_{SN} had a positive effect on Q in seven cases and a negative effect in six cases.

The terminal values for Q and P, averaged over 60 replications (30 for each noise case) for each of the six MGPS-RS variants are presented in Tables 5.8 and 5.9, respectively. In each table, the best result of the six algorithms is enclosed by a rectangle. However, it should be noted, as indicated in the preceding statistical analysis, that the best is not necessarily statistically significant. From Table 5.8, it can be seen that for 18 of 26 problems, the average objective function value of the best result at termination has progressed to within 10% of the difference between the starting and optimal solutions. Table 5.9 illustrates that most average terminal values of P have improved from their original value of 1.0, indicating that the terminal design has moved closer to the optimal solution – an obvious

Table 5.8. **Terminal Value for Performance Measure Q Averaged over 60 Replications (30 for each noise case) – MGPS-RS Algorithms.**

Test Problem	S-MGPS-RIN	S-MGPS-SSM	S-MGPS-SAS	MGPS-RIN	MGPS-SSM	MGPS-SAS
3	$\boxed{0.01308}$	0.02231	0.02333	0.15040	0.23041	0.26814
4	0.11604	$\boxed{0.06449}$	0.09656	0.42984	0.38489	0.37994
5	$\boxed{0.06853}$	0.06905	0.12139	0.13667	0.08224	0.13039
25	$\boxed{0.06845}$	0.07656	0.08417	0.10561	0.10226	0.10697
36	9.184e-5	$\boxed{6.877e-5}$	9.888e-5	14.64e-5	12.78e-5	13.92e-5
105	0.12228	$\boxed{0.10386}$	0.12043	0.40080	0.39563	0.40299
110	0.60609	$\boxed{0.57289}$	0.65849	0.57437	0.60648	0.61715
118	0.07445	0.07561	0.06792	0.09377	$\boxed{0.06721}$	0.08202
224	$\boxed{0.00163}$	0.00178	0.00182	0.00276	0.00211	0.00178
244	0.38432	0.37756	0.60000	0.37394	0.43310	$\boxed{0.36920}$
256	0.00096	$\boxed{0.00050}$	0.00069	0.00129	0.00127	0.00128
275	$\boxed{0.00532}$	0.00552	0.00563	0.00550	0.00564	0.00745
281	0.21216	$\boxed{0.14964}$	0.20407	0.28558	0.22912	0.25370
287	5.662e-4	6.751e-4	7.043e-4	4.223e-4	$\boxed{4.166e-4}$	4.271e-4
288	0.00233	0.00398	0.00396	0.00126	$\boxed{0.00036}$	0.00142
289	1.20395	1.19910	1.17340	1.00405	$\boxed{1.00118}$	1.00119
297	3.510e-4	8.168e-4	18.42e-4	1.463e-4	$\boxed{0.570e-4}$	9.565e-4
300	0.94788	0.92655	$\boxed{0.91767}$	0.98916	0.97512	0.97711
301	0.96703	$\boxed{0.93819}$	1.00422	0.98656	0.94338	1.01613
305	5.032e-9	$\boxed{4.910e-9}$	5.126e-9	5.071e-9	4.939e-9	5.202e-9
314	0.03267	0.03073	$\boxed{0.02279}$	0.03343	0.03318	0.03038
392	0.59389	0.58836	0.59115	0.50274	0.48743	$\boxed{0.48711}$
MVP1	0.00338	0.00228	0.00344	0.00271	$\boxed{0.00140}$	0.00238
MVP2	0.00533	0.00309	0.00316	0.00316	$\boxed{0.00237}$	0.00431
MVP3	0.01481	$\boxed{0.00866}$	0.01270	0.01315	0.00938	0.01224
MVP4	0.00713	$\boxed{0.00497}$	0.00640	0.00613	0.00612	0.00681

sign of convergence. In 24 of 26 cases for Q, and 22 of 26 cases for P, the best solution is produced by an algorithm that uses surrogates, the SSM procedure, or both.

Tables 5.8 and 5.9 also reflect the poor performance of the algorithms for some of the more difficult problems. In particular, problems 300 and 301 are both instances of a gradually sloping quadratic with $n-1$ cross terms so that the function contours are not

Table 5.9. **Terminal Value for Performance Measure P Averaged over 60 Replications (30 for each noise case) – MGPS-RS Algorithms.**

Test Problem	S-MGPS-RIN	S-MGPS-SSM	S-MGPS-SAS	MGPS-RIN	MGPS-SSM	MGPS-SAS
3	1.39540	1.53845	1.45450	$\boxed{0.97642}$	0.98390	0.98830
4	0.43096	$\boxed{0.23574}$	0.35860	0.97671	0.85905	0.72920
5	0.23016	0.27609	0.38620	0.31260	$\boxed{0.22689}$	0.29629
25	$\boxed{0.60901}$	0.69032	0.62041	0.88297	0.87873	0.90482
36	2.771e-4	$\boxed{2.353e-4}$	2.941e-4	3.211e-4	2.804e-4	3.137e-4
105	0.62117	$\boxed{0.61531}$	0.66029	0.93319	0.93494	0.93786
110	0.72689	$\boxed{0.68316}$	0.76373	0.70208	0.72930	0.74529
118	0.54801	0.57414	$\boxed{0.53927}$	0.58954	0.56467	0.60107
224	$\boxed{0.05014}$	0.06133	0.05241	0.06558	0.06494	0.05306
244	0.45466	$\boxed{0.39325}$	0.48229	0.67024	0.69413	0.63578
256	0.10871	$\boxed{0.09211}$	0.09291	0.12091	0.11032	0.10830
275	0.43201	0.49222	0.47172	0.46632	0.44355	$\boxed{0.42695}$
281	0.50573	0.37075	0.48541	0.39984	$\boxed{0.35480}$	0.41369
287	$\boxed{0.36035}$	0.38910	0.38330	0.44449	0.44574	0.44300
288	0.12594	0.13047	0.16748	0.11530	$\boxed{0.07931}$	0.11538
289	1.24475	1.23490	1.20105	1.00400	$\boxed{1.00132}$	1.00130
297	0.02478	$\boxed{0.02168}$	0.03921	0.03703	0.02244	0.04170
300	0.96132	0.96618	$\boxed{0.96129}$	0.99533	0.99445	0.99533
301	0.99200	0.99152	0.99476	0.99179	$\boxed{0.99098}$	0.99400
305	4.51285	$\boxed{4.46560}$	4.52590	4.52985	4.47735	4.54240
314	0.38761	0.41910	$\boxed{0.33489}$	0.41407	0.43805	0.39969
392	0.96754	0.95826	0.96644	0.87721	0.86539	$\boxed{0.86532}$
MVP1	0.10853	0.09774	0.09847	0.09504	$\boxed{0.07842}$	0.09871
MVP2	0.13581	0.10122	$\boxed{0.09606}$	0.09820	0.11828	0.11486
MVP3	0.43704	0.31779	0.38668	0.34565	$\boxed{0.25619}$	0.32271
MVP4	0.46690	0.30877	0.39041	$\boxed{0.28361}$	0.36255	0.37638

along the coordinate axes. This is a hindrance in these experiments because the search directions are set to the axes.

Problem 289 is also a challenging problem with a general nonlinear objective function for which the starting value and optimal value differ only by 0.6963. Hence, the noise has a greater influence for this problem, even in the low noise case, because the noise observed at different candidate designs can dominate the magnitude of the true objective function value difference between those designs. It is also of larger dimension ($n = 30$) than most of the problems. On a smaller scale, problem 244 also presented difficulties for the algorithms. This three-dimensional problem is similar to problem 289 in that the difference between starting and optimal objective function values is small (1.5988); as a result, the best of the MGPS-RS algorithms could only achieve about a 63% reduction in performance measure Q.

From the results of this section, it appears that enough evidence exists to claim that procedure SSM offers performance advantages over the other R&S procedures. However,

Table 5.10. **Number of Switches SW at Termination Averaged over 60 Replications (30 for each noise case) – MGPS-RS Algorithms.**

Test Problem	S-MGPS-SSM	S-MGPS-SAS	MGPS-SSM	MGPS-SAS
3	80,963	155	90,254	222
4	77,202	134	89,675	223
5	80,110	139	90,498	221
25	80,150	219	89,132	292
36	83,714	287	89,718	288
105	83,697	394	88,439	448
110	74,632	365	85,434	666
118	77,749	459	83,910	686
224	90,375	222	90,329	223
244	84,123	226	89,533	292
256	79,485	200	90,017	355
275	89,908	354	87,496	355
281	78,739	421	84,577	672
287	77,695	746	76,441	951
288	60,006	419	79,742	1,012
289	377,400	1,713	405,585	2,144
297	380,265	1,265	408,045	2,061
300	77,073	973	79,512	1,056
301	369,585	2,742	391,645	3,354
305	342,265	2,596	370,750	3,811
314	77,775	141	89,842	224
392	362,100	150	250,635	69
MVP1	87,315	320	87,002	336
MVP2	37,239	142	62,727	237
MVP3	353,845	4,945	416,210	5,126
MVP4	234,737	3,331	284,115	4,156

as discussed in Section 4.1, this discussion is not complete without evaluating the number of switches SW required by the algorithms. Table 5.10 presents the number of cumulative switches required, averaged over 60 replications (30 for each noise case) for the algorithm variants using the SSM and SAS procedures (recall that Rinott's procedure incurs no switching). The table shows that switching for the fully sequential SSM procedure can be quite significant, requiring more switches than SAS by approximately two orders of magnitude on each of the test problems. If used to optimize a real-world system by evaluating a simulation model, this cost must be taken into account before deciding which algorithm variant to use. If the switching cost is negligible relative to the cost of simulation execution, it should not have much impact. However, as Hong and Nelson [59] suggest, switching cost can sometimes exceed sampling costs by orders of magnitude, which could make the use of SSM within MGPS-RS computationally prohibitive.

5.5.2 Comparative Analysis of All Algorithm Implementations

In this subsection, the analysis of the results is extended to the comparison of MGPS-RS with the competing algorithm implementations presented in Section 5.2. The terminal values for Q and P, averaged over 60 replications (30 for each noise case) for each of the four competing algorithms are presented Tables 5.11 and 5.12, which also includes the best result of the MGPS-RS variants. The appropriate MGPS-RS algorithm is listed where, for convenience, the algorithm name is shortened to, for example, S-RIN for "surrogate assisted MGPS with Rinott's procedure" or RIN for "MGPS with Rinott's procedure and no surrogates". In each table, the result that delivered the best average performance measure is enclosed by a rectangle.

The results indicate that, for the continuous-variable problems, the best results are distributed primarily among the two SA procedures. Of the 22 continuous-variable prob-

Table 5.11. **Terminal Value for Performance Measure Q Averaged over 60 Replications (30 for each noise case) – FDSA, SPSA, RNDS, NM, and Best MGPS-RS Algorithms.**

Test Problem	Best of MGPS-RS		FDSA	SPSA	RNDS	NM
3	S-RIN	0.01308	$\boxed{0.00187}$	0.00563	0.04866	–
4	S-SSM	0.06449	$\boxed{0.00018}$	0.00451	0.15692	–
5	S-RIN	0.06853	$\boxed{0.00071}$	0.00098	0.03466	–
25	S-RIN	0.06845	0.09150	0.50503	$\boxed{0.03091}$	–
36	S-SSM	$\boxed{0.069e-3}$	1.260e-3	0.421e-4	7.028e-3	–
105	S-SSM	$\boxed{0.10386}$	0.77604	0.69540	0.61353	–
110	S-SSM	0.57289	0.04767	$\boxed{0.02233}$	0.12655	–
118	SSM	0.06721	$\boxed{0.00245}$ Note 1	0.00840 Note 2	0.32007	–
224	S-RIN	1.628e-3	0.030e-3	$\boxed{0.023e-3}$	1.101e-3	–
244	SAS	0.36920	0.00987	$\boxed{0.00863}$	0.05213	0.50936
256	S-SSM	4.953e-4	$\boxed{0.042e-4}$	16.09e-4	3.347e-4	15.89e-4
275	S-RIN	5.322e-3	$\boxed{0.049e-3}$	0.113e-3	3.286e-3	3.307e-3
281	S-SSM	0.14964	$\boxed{0.06015}$	0.09278	0.21383	0.18178
287	SSM	$\boxed{4.166e-4}$	18.18e-4	4.784e-4	12.38e-4	5.475e-4
288	SSM	3.639e-4	9.443e-4	$\boxed{0.131e-4}$	239.7e-4	211.3e-4
289	SSM	1.00118	0.99157	$\boxed{0.10228}$	1.08630	0.99534
297	SSM	$\boxed{0.057e-3}$	4.024e-3	5.075e-3	6.357e-3	10.13e-3
300	S-SAS	0.91767	0.80106	$\boxed{0.04990}$	0.91633	0.39490
301	S-SSM	0.93819	0.95923	$\boxed{0.56652}$	1.08404	0.76375
305	S-SSM	49.10e-10	$\boxed{2.868e-10}$	4.113e-10	3988.e-10	16.61e-10
314	S-SAS	0.02279	0.00026	$\boxed{0.00013}$	0.00958	0.01749
392	SSM	0.48743	$\boxed{0.00768}$ Note 3	0.00817 Note 4	0.85567	–
MVP1	SSM	$\boxed{0.00140}$	–	–	0.00614	–
MVP2	SSM	$\boxed{0.00237}$	–	–	0.01333	–
MVP3	S-SSM	$\boxed{0.00866}$	–	–	0.02184	–
MVP4	S-SSM	$\boxed{0.00497}$	–	–	0.01110	–

Note 1: 45 of 60 terminal solutions were infeasible with average maximum constraint violation (MCV) of .00226, maximum MCV of 0.02020.
Note 2: 3 of 60 were infeasible with average MCV of .00024, maximum MCV of 0.00032.
Note 3: All 60 were infeasible with average MCV of .02717, maximum MCV of 0.044054.
Note 4: 2 of 60 were infeasible with average MCV of .00315, maximum MCV of 0.00600.

lems, one of these methods claimed the best average performance for each measure Q and P in 17 cases. The FDSA implementation tended to perform better in low dimensions (problems 3, 4, 5, 256, and 275) where SPSA performed better in larger dimensions (problems 288, 289, 300, and 301). This is a tribute to SPSA's efficient technique for estimating the gradient, for which response samples are required at only two design points regardless of problem dimension. The results obtained by SPSA on problems 289, 300, and 301 are fairly remarkable considering the poor performance of the remaining methods, although the Nelder-Mead method enjoyed limited success for quadratic problems 300 and 301.

Although the SA algorithm implementations appear to have superior performance for this group of continuous-variable test problems, MGPS-RS was able to obtain the best average performance on four occasions for each performance measure. This is partly due to the fact that MGPS-RS searches entirely within the feasible region for constrained problems. This was a benefit for problems 25 and 105 because the objective function for these problems can evaluate to a complex number for certain points outside the feasible region. In the MGPS-RS case, infeasible points are easily handled by assigning an arbitrarily large objective function value without sampling. For the SA algorithms, however, the rules governing the perturbation parameter c_k require this parameter to be set relatively large in early iterations (recommended equal to one standard deviation of response sample noise) so that it can gradually decay to zero. If there are restrictive bounds on some variables (as in problem 105) and if infeasible points cannot be evaluated, this leads to a smaller initial setting for c_k. This has a negative impact an gradient accuracy which, to retain stability, necessitates a smaller setting for the initial step size, and therefore slows the convergence since the step size also decays with k.

Another disadvantage of the SA algorithms for constrained problems, as implemented in this research, is that the simple method for correcting infeasible designs suggested in Section 5.2 was unable to avoid infeasibilities for two of the linearly constrained problems (118 and 392). For each replication, the maximum constraint violation (MCV) at termi-

Table 5.12. **Terminal Value for Performance Measure P Averaged over 60 Replications (30 for each noise case) – FDSA, SPSA, RNDS, NM, and Best MGPS-RS Algorithms.**

Test Problem	Best of MGPS-RS		FDSA	SPSA	RNDS	NM
3	RIN	0.97642	0.97415	$\boxed{0.94414}$	1.06560	–
4	S-SSM	0.23574	$\boxed{0.00067}$	0.01677	0.39224	–
5	SSM	0.22689	$\boxed{0.02264}$	0.03144	0.15162	–
25	S-RIN	$\boxed{0.60901}$	0.99261	0.90349	0.86439	–
36	S-SSM	$\boxed{0.00024}$	0.00384	0.00119	0.01972	–
105	S-SSM	$\boxed{0.61531}$	0.99959	0.99666	0.98841	–
110	S-SSM	0.68316	0.18265	$\boxed{0.12051}$	0.28626	–
118	S-SAS	0.53927	0.26615 Note 1	$\boxed{0.21932}$ Note 2	0.73118	–
224	S-RIN	0.05014	0.00438	$\boxed{0.00258}$	0.05936	–
244	S-SSM	0.39325	0.09379	$\boxed{0.08616}$	0.29376	0.61167
256	S-SSM	0.09211	$\boxed{0.00268}$	0.17929	0.08571	0.15597
275	SAS	0.42695	$\boxed{0.07361}$	0.19038	1.25040	0.42242
281	SSM	0.35480	$\boxed{0.03045}$	0.25663	0.90201	0.63811
287	S-RIN	0.36035	$\boxed{0.28593}$	0.33258	0.43073	0.39466
288	SSM	0.07931	0.15083	$\boxed{0.00632}$	0.57399	0.59733
289	SSM	1.00132	0.99194	$\boxed{0.24715}$	1.09515	0.99590
297	S-SSM	$\boxed{0.02168}$	0.63818	0.08092	0.62075	0.52845
300	S-SAS	0.96129	0.94894	$\boxed{0.18877}$	0.96570	0.39665
301	SSM	0.99098	0.99689	0.81008	0.99746	$\boxed{0.73491}$
305	S-SSM	4.46560	$\boxed{0.47072}$	1.26915	39.0515	2.59755
314	S-SAS	0.33489	0.03300	$\boxed{0.02249}$	0.22037	0.30277
392	SSM	0.86539	0.26997 Note 3	$\boxed{0.15424}$ Note 4	0.98951	–
MVP1	SSM	$\boxed{0.07842}$	–	–	0.17562	–
MVP2	S-SAS	$\boxed{0.09606}$	–	–	0.21823	–
MVP3	SSM	$\boxed{0.25619}$	–	–	1.16450	–
MVP4	RIN	$\boxed{0.28361}$	–	–	1.05580	–

Note 1: 45 of 60 terminal solutions were infeasible with average maximum constraint violation (MCV) of .00226, maximum MCV of 0.02020.
Note 2: 3 of 60 were infeasible with average MCV of .00024, maximum MCV of 0.00032.
Note 3: All 60 were infeasible with average MCV of .02717, maximum MCV of 0.044054.
Note 4: 2 of 60 were infeasible with average MCV of .00315, maximum MCV of 0.00600.

nation was recorded, and the average and maximum MCV (over the 60 replications) are annotated in Tables 5.11 and 5.12. An advantage of MGPS-RS algorithms in the presence of linear constraints is that the direction set can be easily updated to incorporate conforming directions, so long as the constraints are not degenerate.

Even though MGPS-RS has some built-in advantages for some of the constrained problems, it is also able to generate competitive results for some of the larger, unconstrained problems (287, 297, and 305). The RNDS and NM methods developed for this research cannot make this claim in general. Each of these methods is outperformed by one of the other methods in every case but one. In problem 25, RNDS generates the best average result for Q and in problem 301, NM generates the best average result for P. Algorithm NM also is at a disadvantage because it cannot be applied unmodified to the constrained problems.

For each of the mixed-variable problems, MGPS-RS outperformed RNDS on both performance measures. This is not surprising since the same conclusion also generally held for the continuous-variable problems.

5.5.3 Termination Criteria Analysis

To complete the analysis of the results, the termination criteria proposed in Section 4.3 are evaluated in this subsection. To facilitate the analysis, various output data were generated in addition to the performance measures described in Section 5.4. At various stages of algorithm execution, the following data were saved to the output file:

- standard deviation of the incumbent design response S_{inc} after the initial stage of s_0 samples,
- indifference zone parameter value δ,
- significance level parameter value α,
- step length parameter value Δ,
- number of iterations completed, and
- number of response samples RS obtained.

Using (4.7) as a guide to predict when the per-iteration sampling requirements might grow rapidly, the analysis was conducted by finding the first point during algorithm progression at which the ratio $\frac{S_{\text{inc}}}{\delta}$ exceeded unity (using $K = 1$ in (4.7)). The iteration at which this occurred, denoted as k', was recorded, as were the values $\Delta_{k'}$, $\alpha_{k'}$, the percent reduction in Q, and the number of response samples RS accumulated. Also computed for the analysis was the percent reduction in Q from iteration k' until termination at iteration k_t, as well as the number of *additional* iterations completed and response samples obtained before termination.

Using algorithm S-MGPS-RIN as a case study for the analysis, the average of these quantities over 30 replications are displayed in Table 5.13 for noise case 1 and Table 5.14 for noise case 2. In the tables, the averages for percent reduction in Q ($\%Q$), number of iterations (Iter.), and response samples (RS) are shown twice: first for the period from initialization to iteration k' ($k \leq k'$), then for the period from k' to termination ($k' < k \leq k_t$). Also listed in the tables is a notional setting for the step length termination scalar Δ_T (4.9), set to the fraction $\frac{1}{100}$ of the initial step length Δ_0.

In each table, the test problems marked with an asterisk indicate those that would have satisfied the termination criteria at iteration k' if the threshold setting for the significance level (4.8) were set to 0.01. For noise case 1, this occurred five times and for noise case 2, eight times. In ten of these 13 cases, excellent progress was made toward the optimal objective function value (97% or higher reduction in Q). Even more telling is that in all cases, very little progress was made from k' until termination while using significantly more samples over fewer iterations. This is, in fact, true for nearly all of the problems which is an indicator that $\frac{S_{\text{inc}}}{\delta}$ may be a useful means for selecting a stopping point.

Table 5.13. **Termination Criteria Analysis for S-MGPS-RIN – Noise Case 1.**

Test				$k \leq k'$			$k' < k \leq k_t$		
Problem	Δ_T	$\Delta_{k'}$	$\alpha_{k'}$	$\%Q$	Iter.	RS	$\%Q$	Iter.	RS
3	.0050	.0060	.0071	99.24	64.3	2,570	0.13	22.3	97,430
* 4	.0025	1.10e-4	.0065	78.82	69.2	2,889	0.72	23.0	97,111
* 5	.0050	.0012	.0075	87.75	62.1	3,464	1.29	18.9	96,536
25	.0200	.0474	.0112	94.37	55.0	3,887	0.43	19.5	96,113
* 36	.0100	2.04e-5	.0092	99.99	75.3	5,111	0.00	26.1	94,889
105	.0025	.0865	.0415	82.73	40.9	5.860	0.89	20.5	94,140
110	.0010	.0389	.0152	5.47	39.8	11,556	8.76	10.0	88,444
118	.0400	.7401	.0385	86.84	44.1	9,110	1.78	16.8	90,890
* 224	.0050	4.28e-5	.0089	99.75	87.1	4,481	0.00	27.3	95,519
244	.0200	.0223	.0100	45.95	62.9	4,794	0.58	22.4	95,206
256	.0100	.0200	.0100	99.84	43.8	6,380	0.03	12.2	93,620
* 275	.0100	.0042	.0092	99.25	84.8	7,912	0.01	23.1	92,088
281	.0050	.1969	.0235	52.87	47.8	11,065	10.46	13.7	88,935
287	.0100	.3490	.0749	99.78	35.5	18,386	0.14	15.2	81,614
288	.0100	.4708	.0324	98.99	31.6	29,702	0.55	7.0	70,298
289	.0010	.0642	.0128	-20.46	67.9	57,823	0.25	16.1	442,177
297	.0200	.2667	.0852	97.41	22.5	18,076	2.53	36.1	481,924
300	.0010	.0652	.0391	-6.18	43.4	20,558	1.00	10.9	79,442
301	.0200	.7875	.0698	-12.19	23.9	35,159	8.80	10.4	464,841
305	.0200	.1760	.0451	100.0	28.2	295,210	0.00	1.1	204,790
314	.0025	.0027	.0080	94.82	61.8	2,836	0.45	20.6	97,164
392	.1000	2.8757	.0692	37.76	28.1	3,444	3.60	26.0	496,556
MVP1	.0050	.0776	.0119	99.45	39.2	17,444	0.12	13.3	82,557
MVP2	.0050	.2408	.0593	99.22	24.6	18,264	0.02	1.6	81,736
MVP3	.0050	.0860	.0284	97.36	201.3	259,358	0.03	26.3	240,642
MVP4	.0050	.2776	.0924	98.68	98.9	129,302	0.02	6.5	370,698

It should be noted, however, that for some problems, particularly in noise case 1, some mildly significant improvement was still possible after iteration k' (*e.g.*, problems 110, 281, and 301). In each of these cases, the average step length had not been reduced dramatically from its initial setting, indicating that the algorithm may still have been making many successful moves through the design space. In these situations, it would be advantageous to have a parameter update strategy that monitored the decay rate of Δ_k and adjusted the decay rate of α_k and δ_k accordingly. That is, if Δ_k is decaying slowly then so should α_k

Table 5.14. **Termination Criteria Analysis for S-MGPS-RIN – Noise Case 2.**

Test				$k \le k'$			$k' < k \le k_t$		
Problem	Δ_T	$\Delta_{k'}$	$\alpha_{k'}$	$\%Q$	Iter.	RS	$\%Q$	Iter.	RS
3	.0050	.0060	.0071	99.24	64.3	2,570	0.13	22.3	97,430
* 4	.0025	2.51e-5	.0068	97.09	70.1	2,563	0.16	26.0	97,437
5	.0050	.0084	.0067	96.68	62.3	2,877	0.57	19.4	97,123
* 25	.0200	.0086	.0048	91.44	71.0	5,054	0.07	18.4	94,946
* 36	.0100	1.35e-5	.0074	99.99	78.8	5,722	0.00	24.6	94,278
105	.0025	.1360	.0025	91.48	74.6	20,308	0.45	9.2	79,692
110	.0010	.0331	.0053	60.48	49.6	19,046	4.07	6.6	80,954
118	.0400	.2810	.0031	95.70	77.8	25,140	0.78	8.1	74,860
* 224	.0050	5.69e-5	.0073	99.92	90.6	4,164	0.00	28.9	95,836
244	.0200	.0243	.0074	75.25	58.4	4,002	1.36	19.1	95,998
256	.0100	.0111	.0077	99.93	45.6	5,232	0.02	12.8	94,768
* 275	.0100	.0027	.0078	99.67	88.2	8,345	0.00	22.5	91,655
281	.0050	.0223	.0060	93.63	50.3	17,368	0.61	7.1	82,632
287	.0100	.0193	.0020	99.97	99.0	79,485	0.00	1.4	20,515
288	.0100	.1959	.0074	99.99	45.8	25,781	0.00	4.6	74,219
289	.0010	.0519	.0067	-20.86	77.2	72,421	0.27	13.3	427,579
* 297	.0200	.0164	.0066	99.99	68.8	76,727	0.01	12.3	423,273
300	.0010	.0374	.0023	15.49	99.8	79,607	0.11	1.7	20,393
301	.0200	.0891	.0014	9.84	84.5	319,703	0.15	4.2	180,297
* 305	.0200	.0164	.0023	100.0	73.9	432,329	0.00	0.2	67,671
* 314	.0025	.0024	.0066	97.99	63.0	3,138	0.21	20.1	96,862
392	.1000	.8835	.0009	39.45	101.9	52,511	0.42	21.1	447,489
MVP1	.0050	.0321	.0076	99.72	47.5	9,216	0.03	20.7	90,784
MVP2	.0050	.2105	.0174	99.67	19.8	16,514	0.02	4.7	83,486
MVP3	.0050	.0080	.0042	99.64	302.3	286,778	0.01	29.1	213,222
MVP4	.0050	.0532	.0076	99.85	183.7	298,393	0.02	21.0	201,607

and δ_k to allow the search to continue exploring the design space aggressively before the sampling requirements increase to prohibitive levels.

5.5.4 Summary of the Analysis

The analytical results of this section provide enough evidence to draw some conclusions regarding the performance of the algorithms on the test problems considered in these experiments. First, the use of surrogates and/or the SSM R&S procedure appears to have a positive effect on algorithm performance in many cases, although the importance of trading off switching costs with algorithm progress was illustrated. Secondly, the stochastic approximation algorithms generally perform better for problems with continuous variables only, although constrained problems can present these methods with some difficulties. Finally, the proposed MGPS-RS termination criteria seem to offer a valid mechanism to establish some algorithm stopping decision rules, particularly if the algorithms are modified to allow adaptive update strategies for the R&S parameters.

Chapter 6 - Conclusions and Recommendations

A new class of algorithms for solving mixed-variable stochastic optimization problems has been presented. It further generalizes the class of generalized pattern search algorithms to noisy objective functions, using ranking and selection statistical procedures in the selection of new iterates. A rigorous analysis proves new convergence theorems, demonstrating that a subsequence of iterates converges with probability one to a stationary point appropriately defined in the mixed-variable domain. Additionally, advanced algorithm options using modern R&S procedures, surrogate functions, and termination criteria, that provide computational enhancements to the basic algorithm, have been developed and implemented. Computational tests reveal that the advanced options can indeed improve performance, allowing a performance comparison to popular methods from the stochastic optimization literature. The original contributions of this dissertation research are summarized in Section 6.1 and future research directions are proposed in Section 6.2.

6.1 Contributions

The primary contribution of this research is that it develops the first convergent algorithm for numerically solving stochastic optimization problems over mixed-variable domains. Although convergent algorithms that apply to continuous-only domains (*e.g.*, stochastic approximation) or discrete-only domains (*e.g.*, random search) have been devised, the algorithms presented in this dissertation bridge the gap between these two domain types, illustrating enhanced generality relative to existing methods. MGPS-RS algorithms are further generalized by their ability to readily handle problems with variable bounds and/or a finite number of linear constraints in addition to unconstrained problems.

Although the convergence theory in Section 3.7 builds upon existing pattern search theory [1,13,79–81,143,146], additional mathematical constructs were required to establish new convergence results. In particular, the conditions placed on R&S parameters α_r and δ_r and the resulting proofs of Lemmas 3.13 and 3.14 using the Borel-Cantelli lemma are original developments. The remaining lemmas and theorems extend existing theory by establishing convergence in probabilistic terms appropriate for the stochastic setting.

Another important contribution of this research is the development of a viable strategy for employing surrogate functions to augment the search. Although surrogate search strategies for pattern search are not without precedent [30–32, 39, 86, 127, 145, 147], this research is the first to introduce kernel regression within a pattern search framework, the first to apply surrogate-assisted pattern search algorithms in a stochastic setting, and the first to make use of surrogates in solving MVP problems.

A third contribution is the development of effective termination criteria by combining the traditional step length thresholding criterion with additional rules intended to avoid unnecessary sampling. This strategy expresses the *practical* difference between two candidates at termination, reflected in the indifference zone parameter, in terms of the standard deviation of the response samples, providing a heuristic means to predict when sampling requirements will dramatically increase. At the same time, it also ensures that the probability of selecting the best candidate in the terminal iteration meets a minimum threshold. Such a method is important using sampling based methods to provide a means to impose controls on potentially excessive sampling requirements.

A final contribution of this research is the computational study. The literature consists of a limited number of such studies, primarily in the evaluation of direct search methods (*e.g.*, [6, 23, 62]), yet these studies are restricted to unconstrained test problems and moderate dimension, typically $n = 20$ or less. In the study presented in this dissertation, substantial effort was directed toward the selection of a wide range of problem types and algorithm implementations. The mixture of objective function types, constraint types, and range of problem dimension is perhaps the most extensive collection to be tested in a stochastic setting.

6.2 Future Research

The work presented in this dissertation can be extended in many directions. Suggestions for future research can be generally organized into two categories: modifications to the algorithmic framework; and extensions of the framework to a broader class of stochastic optimization problems. These broad categories are discussed in the following subsections.

6.2.1 Modifications to Existing Search Framework

Adaptive Parameter Updates. It was briefly mentioned in Sections 4.3 and 5.5.3 that adaptive methods for updating algorithm parameters α_r and δ_r may have the potential to improve algorithm performance. In particular, if the parameters are decreased too aggressively, then the algorithm increases the precision of the iterate selection decision earlier in the iteration sequence than if the decay rate is slower. This can adversely impact performance because the sampling requirements may increase prematurely while the search is still actively moving through the design space, potentially slowing progress toward optimality. It would be beneficial to investigate adaptive parameter updates based on knowledge gained during the search. Candidate strategies might include monitoring the rate of decay of the step length parameter Δ_k or the ratio of successful iterations to unsuccessful iterations, and using the information gained to adjust the decay rate of α_r and δ_r. Additionally, in this research these parameters are represented as geometric sequences, but alternative sequences may lead to better performance as long as the conditions required for algorithm convergence are retained (assumption **A6** in Section 3.7).

Selecting the Number of Design Sites. In the computational evaluation, the number of design sites used to build the original surrogate function(s) was fixed, regardless of the problem dimension. For the larger problems, it is possible that this led to greater inaccuracies between the surrogate and true surfaces and the subsequent ineffective SEARCH steps observed in Section 5.5.1. A worthy endeavor would be to design a rule to link the number of original design sites to problem dimension. As the dimension grows, simple procedures may include increasing the strength of the LHS design and/or the number of intervals p that divide each dimension.

Alternative Kernel Functions. In this research, Gaussian kernels were used to build the surrogate functions, but alternative mound-shaped kernels may offer advantages over Gaussians. An example is the Epanechnikov kernel with parabolic shape, described in [43] and [55, pp. 25-28]. This kernel takes a value of zero outside a fixed interval, which can lead to numerical benefits over a Gaussian kernel. The Gaussian kernel takes on very small values for points sufficiently far from all design sites, which can cause numerical underflow on a computer [55, p. 25]. There may be additional benefits related to surrogate accuracy that can be realized by using alternative kernels.

Alternative Surrogate Families. Another modification to the surrogate-based approach is to replace kernel regression surfaces with an alternative family of surrogates. In previous studies coupling surrogates with pattern search [30–32, 39, 86, 127, 145, 147], kriging or interpolating splines were used as the methods to approximate the response surface. Many other methods, such as traditional polynomial regression via least squares or the use of artificial neural networks, may be tried that might lead to improvements in algorithm performance.

Improving the Efficiency of Searching the Surrogate. Additional efficiencies may be realized by modifying how the search of the surrogate is conducted. In the algorithm implementation described in Section 4.4.2, search of the surrogate surface is conducted via a pattern search on the current mesh. However, it is also pointed out that for a very fine mesh and a large number of design sites, this can become costly. Potential efficiencies may be gained by replacing the pattern search with an alternative procedure, perhaps a gradient or quasi-Newton search by deriving an analytical expression for the merit function gradient, and then mapping the resultant point to the nearest mesh point. Another approach may be to reduce the number of points that must be evaluated in the surrogate function by using *k-means clustering* [41, pp. 526-528] to group the design sites into a smaller set of points. Using this approach, the design sites are grouped into k clusters and the mean of each cluster replaces all points in that cluster when evaluating the surrogate function. Many iterative procedures exist (see [5]) to determine the number and location of the clusters necessary to satisfy some predetermined criterion.

Balancing Sampling and Switching Costs. The analysis of Section 5.5 revealed that, although the SSM procedure appears to achieve better solutions over a fixed sampling budget, it can require a large number of switches between candidate designs. As a means to balance the cost of sampling with the cost of switching, a simple modification may be to implement the *minimum switching sequential procedure*, a R&S procedure recently developed by Hong and Nelson [59], and evaluate its performance relative to the R&S procedures implemented in this dissertation research. Hong and Nelson's procedure, a two-stage sampling procedure, uses the same number of switches as two-stage procedures when additional samples are required for all candidates after the initial stage, but still maintains sequential sampling.

Expanded Computational Testing. The computational evaluation of Chapter 5 provides valuable numerical experience and insights for MGPS-RS algorithms and their performance relative to existing methods. However, an expanded testing program could further enhance the understanding of when MGPS-RS might enjoy success and where additional deficiencies reside. This testing should be broadened to more problems and may consider some of the recommendations presented in the preceding paragraphs. Additional value would be added with some case studies that applied the MGPS-RS algorithms to the optimization of some real-world stochastic systems for which representative simulation models exist.

6.2.2 Extensions to Broader Problem Classes

Relaxing the Smoothness Assumption. A restrictive assumption of the convergence analysis is that the true objective function is continuously differentiable with respect to the continuous variables when the discrete variables are fixed. Applying the Clarke calculus [35] in the deterministic setting, Audet and Dennis [14] relax this assumption and present a hierarchy of convergence results where the strength of the results depend on local smoothness properties. A worthy research avenue would be to further extend the results for the stochastic setting in the context of the MGPS-RS framework.

Nonlinear Constraints. In this research, the constraints are restricted to bound and linear constraints only. A worthwhile extension to MGPS-RS algorithms would be to make them applicable to nonlinear constraints also, perhaps by adapting tools from any of the three pattern search methods applied to nonlinearly constrained deterministic problems discussed in Section 2.2.1: the augmented Lagrangian approach of Lewis and Torczon [82], the filter method of Audet and Dennis [16], and the Mesh Adaptive Direct Search (MADS) algorithm of Audet and Dennis [15]. In particular, the MADS algorithm extends pattern search by generating mesh directions that become dense in the limit. However, as these directions become dense, their number becomes unbounded. Since this is prohibitive in practice, an implementable instance is provided in [15], in which positive spanning directions are randomly selected at each iteration. The cost of doing so, however, is the weaker condition of convergence with probability 1 to a stationary point. Extending any of the deterministic algorithms [15, 16, 82] to the stochastic setting results in the weaker convergence results, but nothing is lost with MADS, since its convergence result is already in probabilistic terms.

Multiple Responses. In this research, the target problem class contains only a single system response output, the minimization of which is the objective of the optimization task. This problem class can be broadened to problems that have multiple response outputs. Depending on the objectives of the optimization problem, the additional responses can be considered in two different ways: (a) as additional constraints, or (b) as additional objectives. In the first case, the additional responses may be constrained to a specified performance range. Since these responses may not be linear, any of the techniques suggested in the preceding paragraph for handling nonlinear constraints could be employed. However, the stochastic nature of the new constraints may require additional assumptions to ensure sound theoretical convergence results.

In the second case, the target problem becomes one with multiple objectives. For multi-objective stochastic optimization problems, specialized statistical procedures are needed to select iterates and retain the rigor of the selection. An approach for simulation optimization using direct search is suggested in [90], employing Hotelling's T^2 procedure. This approach consists of two testing phases to compare the incumbent with a single candidate, a first phase consisting of an all-pairwise two-sample comparison of means of all responses followed by a second phase of a two-sample comparison of means on a weighted sum of all responses. After the first phase, if at least one response mean of the candidate is significantly deficient or all response means are statistically insignificant, the candidate is rejected. If all response means are significantly improved, then the candidate is accepted. However, if at least one response mean is significantly improved while at least one is statistically insignificant, then the second phase is conducted where the candidate is accepted if the weighted sum function is significantly improved and rejected otherwise. This approach is extended in [89] to account for correlation between the different responses. Employing such a method within the pattern search framework in lieu of an R&S procedure to select new iterates may be a worthwhile research area for multi-objective stochastic optimization. Alternatively, some limited work in indifference-zone R&S procedures has been done to extend these techniques to multiple responses (see [140, pp. 139-140] for a brief review), which would coincide more closely with the framework presented in this dissertation. It would be useful to explore iterative use of these methods within MGPS-RS for multi-objective stochastic optimization.

APPENDIX A - Test Problem Details

This Appendix provides the details of the twenty-two continuous-variable test problems used in the computational evaluation. The problem numbers are as assigned in the publications from which they were selected [58, 122]. Each of the problems is classified according to the category combination defined in Section 5.3. The classification scheme has a three letter designator: the first letter is the objective function type (Q - quadratic, S - sum of squares, P - generalized polynomial, G - general nonlinear); the second letter is the constraint information (U - unconstrained, B - bounds only, L - linear constraints and bounds); and the third letter designates problem size (S - small, M - medium, L - large). Each problem listing also includes the number of variables, the number of bounds, the number of linear constraints, the objective functional form, the starting point, the optimal solution, and the form of the constraints.

Problem 3

Category combination:	QBS
Number of variables:	2
Number of bounds:	1
Number of linear constraints:	0
Objective function:	$f(x) = x_2 + 10^{-5}(x_2 - x_1)^2$
Bounds:	$0 \leq x_2$
Starting point:	$x = (10, 1),\ f(x) = 1.0081$
Optimal solution:	$x^* = (0, 0),\ f(x^*) = 0$

Problem 4

Category combination:	GBS
Number of variables:	2
Number of bounds:	2
Number of linear constraints:	0
Objective function:	$f(x) = \frac{1}{3}(x_1+1)^3 + x_2$
Bounds:	$1 \leq x_1, \quad 0 \leq x_2$
Starting point:	$x = (1.125, 0.125),\ f(x) = 3.323568$
Optimal solution:	$x^* = (1, 0),\ f(x^*) = \frac{8}{3}$

Problem 5

Category combination:	GBS
Number of variables:	2
Number of bounds:	4
Number of linear constraints:	0
Objective function:	$f(x) = \sin(x_1+x_2) + (x_1-x_2)^2 - 1.5x_1 + 2.5x_2 + 1$
Bounds:	$-1.5 \leq x_1 \leq 4, \quad -3 \leq x_2 \leq 3$
Starting point:	$x = (0, 0),\ f(x) = 1$
Optimal solution:	$x^* = (-\frac{\pi}{3}+\frac{1}{2}, -\frac{\pi}{3}-\frac{1}{2}),\ f(x^*) = -\frac{1}{2}\sqrt{3} - \frac{\pi}{3}$

Problem 25

Category combination:	SBS
Number of variables:	3
Number of bounds:	6
Number of linear constraints:	0
Objective function:	$f(x) = \sum_{i=1}^{99} (f_i(x))^2$
	where $f_i(x) = -.01i + \exp\left(-\frac{1}{x_1}(u_i - x_2)^{x_3}\right)$
	and $u_i = 25 + (-50\ln(.01i))^{2/3}, \ i = 1, \ldots, 99$
Bounds:	$0.1 \le x_1 \le 100, \ 0 \le x_2 \le 25.6, \ 0 \le x_3 \le 5$
Starting point:	$x = (100, 12.5, 3), \ f(x) = 32.835$
Optimal solution:	$x^* = (50, 25, 1.5), \ f(x^*) = 0$

Problem 36

Category combination:	PLS
Number of variables:	3
Number of bounds:	6
Number of linear constraints:	1
Objective function:	$f(x) = -x_1x_2x_3$
Bounds:	$0 \le x_1 \le 20, \ 0 \le x_2 \le 11, \ 0 \le x_3 \le 42$
Starting point:	$x = (10, 10, 10), \ f(x) = -1000$
Optimal solution:	$x^* = (20, 11, 15), \ f(x^*) = -3300$
Constraint:	$x_1 + 2x_2 + 2x_3 \le 72$

Problem 105

Category combination:	GLS
Number of variables:	8
Number of bounds:	16
Number of linear constraints:	1

Objective function:

$$f(x) = \sum_{i=1}^{235} \ln\left((a_i(x) + b_i(x) + c_i(x))/\sqrt{2\pi}\right)$$

where for $i = 1, \ldots, 235$,

$$a_i(x) = \tfrac{x_1}{x_6} \exp\left(-(y_i - x_3)^2/2x_6^2\right),$$

$$b_i(x) = \tfrac{x_2}{x_7} \exp\left(-(y_i - x_4)^2/2x_7^2\right),$$

$$c_i(x) = \tfrac{1-x_2-x_1}{x_8} \exp\left(-(y_i - x_5)^2/2x_8^2\right),$$

and y_i is as defined in Table that follows

Bounds:

$0.001 \le x_i \le 0.499$, $i = 1, 2$,
$100 \le x_3 \le 180$, $120 \le x_4 \le 210$, $170 \le x_5 \le 240$,
$5 \le x_i \le 25$, $i = 6, 7, 8$

Starting point:

$x = (.1, .2, 100, 125, 175, 11.2, 13.2, 15.8)$,
$f(x) = 1297.6693$

Optimal solution:

$x^* = (.4128928, .4033526, 131.2613, 164.3135,$
$217.4222, 12.28018, 15.77170, 20.74682)$,
$f(x^*) = 1138.416240$

Constraint:

$x_1 + x_2 \le 1$

Additional Data (y_i) for Problem 105.

i	y_i	i	y_i	i	y_i
1	95	102–118	150	199–201	200
2	105	119–122	155	202–204	205
3–6	110	123–142	160	205–212	210
7–10	115	143–150	165	213	215
11–25	120	151–167	170	214–219	220
26–40	125	168–175	175	220–224	230
41–55	130	176–181	180	225	235
56–68	135	182–187	185	226–232	240
69–89	140	188–194	190	233	245
90–101	145	195–198	195	234–235	250

Problem 110

Category combination:	GBM
Number of variables:	10
Number of bounds:	20
Number of linear constraints:	0
Objective function:	$f(x) = \sum_{i=1}^{10} \left[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2\right] - \left(\prod_{i=1}^{10} x_i\right)^{0.2}$
Bounds:	$2.001 \le x_i \le 9.999,\ i = 1, \ldots, 10$
Starting point:	$x = (9, \ldots, 9),\ f(x) = -43.134337$
Optimal solution:	$x^* = (9.35025655, \ldots, 9.35025655),$ $f(x^*) = -45.77846971$

Problem 118

Category combination:	QLM
Number of variables:	15
Number of bounds:	30
Number of linear constraints:	29
Objective function:	$f(x) = \sum_{i=1}^{4}(2.3x_{3i+1} + .0001x_{3i+1}^2 + 1.7x_{3i+2} + .0001x_{3i+2}^2 + 2.2x_{3i+3}^2 + .00015x_{3i+3}^2)$
Bounds:	$8 \le x_1 \le 21$, $43 \le x_2 \le 57$, $3 \le x_3 \le 16$, $0 \le x_{3i+1} \le 90$, $0 \le x_{3i+2} \le 120$, $0 \le x_{3i+3} \le 60$, $i = 1, 2, 3, 4$
Starting point:	$x = (20, 55, 15, 20, 60, 20, 20, 60, 20, 20, 60, 20, 20, 60, 20)$, $f(x) = 769.8400$
Optimal solution:	$x^* = (8, 49, 3, 1, 56, 0, 1, 63, 6, 3, 70, 12, 5, 77, 18)$ $f(x^*) = 556.2726$
Constraints:	$l \le Ax \le u$, where

$$l = [-7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, -7, 60, 50, 70, 85, 100]^T,$$

$$u = [6, 6, 6, 6, 7, 7, 7, 7, 6, 6, 6, 6, \infty, \infty, \infty, \infty, \infty]^T, \text{ and}$$

$$A = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Problem 224

Category combination:	QLS
Number of variables:	2
Number of bounds:	4
Number of linear constraints:	4
Objective function:	$f(x) = 2x_1^2 + x_2^2 - 48x_1 - 40x_2$
Bounds:	$0 \le x_1 \le 6,\ \ 0 \le x_2 \le 6$
Starting point:	$x = (0.1, 0.1),\ f(x) = -8.77$
Optimal solution:	$x^* = (4, 4),\ f(x^*) = -304$
Constraints:	$l \le Ax \le u$, where

$$l = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 3 \\ 1 & 1 \end{bmatrix}, \text{ and } \quad u = \begin{bmatrix} 18 \\ 8 \end{bmatrix}$$

Problem 244

Category combination:	SUS
Number of variables:	3
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = \sum_{i=1}^{10} [\exp(-x_1 z_i) - x_3 \exp(-x_2 z_i) - y_i]^2$
	where $y_i(x) = \exp(-z_i) - 5\exp(-10z_i)$
	and $z_i = 0.1i, \;\; i = 1, \ldots, 10$
Starting point:	$x = (1, 2, 1), \; f(x) = 1.59884$
Optimal solution:	$x^* = (1, 10, 5), \; f(x^*) = 0$

Problem 256

Category combination:	PUS
Number of variables:	4
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4$
	$+10(x_1 - x_4)^4$ (Powell function)
Starting point:	$x = (3, -1, 0, 1), \; f(x) = 215$
Optimal solution:	$x^* = (0, 0, 0, 0), \; f(x^*) = 0$

Problem 275

Category combination:	QUS
Number of variables:	4
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = x^T Q x$
	where $Q(i,j) = \frac{1}{i+j-1}$ (4×4 Hilbert Matrix)
Starting point:	$x = (-4, -2, -1.333, -1),\ f(x) = 33.9651$
Optimal solution:	$x^* = (0,0,0,0),\ f(x^*) = 0$

Problem 281

Category combination:	GUM
Number of variables:	10
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = \left(\sum_{i=1}^{10} i^3 (x_i - 1)^2 \right)^{1/3}$
Starting point:	$x = (0, \ldots, 0),\ f(x) = 14.4624$
Optimal solution:	$x^* = (1, \ldots, 1),\ f(x^*) = 0$

Problem 287

Category combination:	PUM
Number of variables:	20
Number of bounds:	0
Number of linear constraints:	0

Objective function:

$$f(x) = \sum_{i=1}^{5} \big[100(x_i^2 - x_{i+5})^2 + (x_i - 1)^2$$
$$+90(x_{i+10}^2 - x_{i+15})^2 + (x_{i+10} - 1)^2$$
$$+10.1((x_{i+5} - 1)^2 + (x_{i+15} - 1)^2)$$
$$+19.8(x_{i+5} - 1)(x_{i+15} - 1)\big]$$

Starting point: $x_i = -3$, $i = 1, \ldots, 5, 11, \ldots, 15,$
$x_i = -1$, $i = 6, \ldots, 10, 16, \ldots, 20$
$f(x) = 95960$

Optimal solution: $x^* = (1, \ldots, 1)$, $f(x^*) = 0$

Problem 288

Category combination:	SUM
Number of variables:	20
Number of bounds:	0
Number of linear constraints:	0

Objective function:

$$f(x) = \sum_{i=1}^{5} (x_i + 10x_{i+5})^2 + 5(x_{i+10} - x_{i+15})^2$$
$$+(x_{i+5} - 2x_{i+10})^4 + 10(x_i - x_{i+15})^4$$

Starting point: $x_i = 3$, $i = 1, \ldots, 5$, $x_i = -1$, $i = 6, \ldots, 10,$
$x_i = 0$, $i = 11, \ldots, 15$, $x_i = 1$, $i = 16, \ldots, 20,$
$f(x) = 1075$

Optimal solution: $x^* = (0, \ldots, 0)$, $f(x^*) = 0$

Problem 289

Category combination:	GUL
Number of variables:	30
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = 1 - \exp\left[-\frac{1}{60}\sum_{i=1}^{30} x_i^2\right]$
Starting point:	$x = (-1.03, 1.07, -1.10, 1.13, -1.17, 1.20, -1.23, 1.27, -1.30, 1.33, -1.37, 1.40, -1.43, 1.47, -1.50, 1.53, -1.57, 1.60, -1.63, 1.67, -1.70, 1.73, -1.77, 1.80, -1.83, 1.87, -1.90, 1.93, -1.97, 2.00)$, $f(x) = 0.696313$
Optimal solution:	$x^* = (0, \ldots, 0)$, $f(x^*) = 0$

Problem 297

Category combination:	SUL
Number of variables:	30
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = \sum_{i=1}^{29} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]$ (Rosenbrock banana function)
Starting point:	$x = (-1.2, 1, \ldots, -1.2, 1)$, $f(x) = 7139$
Optimal solution:	$x^* = (1, \ldots, 1)$, $f(x^*) = 0$

Problem 300

Category combination:	QUM
Number of variables:	20
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = x^T Q x - 2x_1$

$$\text{where } Q = \begin{bmatrix} 1 & -1 & 0 & \dots & & 0 \\ -1 & 2 & -1 & 0 & & \vdots \\ 0 & -1 & 2 & -1 & 0 & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ & & 0 & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{bmatrix}$$

Starting point:	$x = (0, \dots, 0),\ f(x) = 0$
Optimal solution:	$x^* = (20, 19, 18, \dots, 2, 1),\ f(x^*) = -20$

Problem 301

Category combination:	QUL
Number of variables:	50
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = x^T Q x - 2x_1$

$$\text{where } Q = \begin{bmatrix} 1 & -1 & 0 & \dots & & 0 \\ -1 & 2 & -1 & 0 & & \vdots \\ 0 & -1 & 2 & -1 & 0 & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ & & 0 & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{bmatrix}$$

Starting point:	$x = (0, \dots, 0),\ \ f(x) = 0$
Optimal solution:	$x^* = (50, 49, 48, \dots, 2, 1),\ f(x^*) = -50$

Problem 305

Category combination:	PUL
Number of variables:	100
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = \sum_{i=1}^{100} x_i^2 + \left(\sum_{i=1}^{100} \frac{1}{2} i \ x_i \right)^2 + \left(\sum_{i=1}^{100} \frac{1}{2} i \ x_i \right)^4$
Starting point:	$x = (0.1, \dots, 0.1),\ f(x) = 4064923200$
Optimal solution:	$x^* = (0, \dots, 0),\ f(x^*) = 0$

Problem 314

Category combination:	GUS
Number of variables:	2
Number of bounds:	0
Number of linear constraints:	0
Objective function:	$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + \frac{0.04}{g(x)} + \frac{(h(x))^2}{0.2}$,
	where $g(x) = -\frac{1}{4}x_1^2 - x_2^2 + 1$,
	and $h(x) = x_1 - 2x_2 + 1$
Starting point:	$x = (2, 2),\ f(x) = 5.99$
Optimal solution:	$x^* = (1.789, 1.374),\ f(x^*) = 0.169040$

Problem 392

Category combination:	QLL
Number of variables:	30
Number of bounds:	45
Number of linear constraints:	30

Objective function:

$$f(x) = \sum_{i=1}^{5}\sum_{j=1}^{3}\Big[(r_{1ji} - k_{Aji})x_{3(i-1)+j} - r_{2ji}x_{3(i-1)+j}^2$$

$$-(k_{1ji} + k_{Pji})x_{12+3i+j}$$

$$-(k_{3ji} + k_{L1ji})(x_{12+3i+j} - x_{j+3i-3})^2$$

$$-k_{L2ji}\sum\nolimits_{\ell=1}^{i}(x_{12+j+3\ell} - x_{j-3+3\ell})\Big]$$

where r_{1ji}, r_{2ji}, k_{Aji}, k_{1ji}, k_{Pji}, k_{3ji}, k_{L1ji}, and k_{L2ji} are as defined in Table that follows

Bounds:

$0 \le x_1 \le 100$, $0 \le x_2 \le 280$, $0 \le x_3 \le 520$,
$0 \le x_4 \le 180$, $0 \le x_5 \le 400$, $0 \le x_6 \le 400$,
$0 \le x_7 \le 220$, $0 \le x_8 \le 450$, $0 \le x_9 \le 500$,
$0 \le x_{10} \le 150$, $0 \le x_{11} \le 450$, $0 \le x_{12} \le 630$,
$0 \le x_{13} \le 100$, $0 \le x_{14} \le 400$, $0 \le x_{15} \le 600$,
$x_i \ge 0$, $i = 16, \dots, 30$

Starting point:

$x = (80, 50, 370, 100, 150, 200, 100, 250, 400, 50,$
$200, 500, 50, 200, 500, 100, 120, 410, 120, 190,$
$190, 60, 240, 370, 130, 100, 510, 30, 250, 510),$[3]
$f(x) = -845999$

Optimal solution:

$x^* = (99.99, 142.22, 519.88, 136.74, 103.47, 399.99,$
$191.70, 1.56, 500, 143.43, 82.39, 629.82,$
$99.92, 125.22, 600, 101.85, 142.25, 519.88,$
$144.58, 105.73, 409.59, 182.01, 29.34, 490.52,$
$143.43, 52.43, 629.70, 99.92, 125.12, 600),$
$f(x^*) = -1693551.668$

Constraints:

$l \le Ax \le u$, where
$l_i = -\infty$, $i = 1, \dots, 15$, $l_i = 0$, $i = 16, \dots, 30$,
$u_i = 170$, $i = 1, 2, 4, 5, 7, 8, 10, 11, 13, 14$,
$u_i = 180$, $i = 3, 6, 8, 12, 15$, $u_i = \infty$, $i = 16, \dots, 30$,

[3]Starting point was modified from published version to make it feasible.

$$A = \left[\begin{array}{cc} A_1 & A_2 \end{array} \right],$$

$$A_1 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1
\end{bmatrix},$$

and

$$A_2 = \begin{bmatrix}
.6 & .4 & .1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
.3 & .1 & .12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
.36 & .08 & .06 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .6 & .4 & .1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .3 & .1 & .12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .36 & .08 & .06 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .4 & .1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .3 & .1 & .12 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .36 & .08 & .06 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .6 & .4 & .1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & .1 & .12 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .36 & .08 & .06 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .6 & .4 & .1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & .1 & .12 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .36 & .08 & .06 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}$$

Additional Data for Problem 392.

	$j = 1$					$j = 2$					$j = 3$				
	i					i					i				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
r_{1ji}	1000			1100		520			600		910		1000		
r_{2ji}	0.3					0.1					0.2				
k_{Aji}	120	150	150	170	170	65		80			105		120		
k_{1ji}	150			170		75			90		140			150	
k_{Pji}	160			180		75			90		140			150	
k_{3ji}	.02	.2	.25			.01	.1	.1	.15		.015	.15			
k_{L1ji}	.005	.05	.06			.005	.05	.06			.005	.05	.06		
k_{L2ji}	80		100			45			50		75		90		

APPENDIX B - Test Result Data

Some additional details from the test results are presented in this appendix. Section B.1 presents a series of charts that shows the iteration history of the algorithms for all test problems. Section B.2 provides the data to support the ANOVA and nonparametric procedures of Section 5.5.1.

B.1 Iteration History Charts

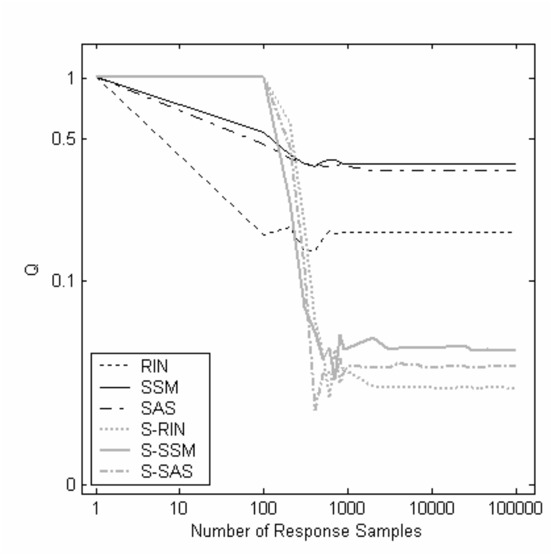
To give a visual perspective of algorithm progression, a series of graphs are displayed on the pages that follow that plot performance measure Q, averaged over 30 replications, versus the number of response samples obtained for each algorithm, noise case, and test problem. The graphs are shown on log scales so that the progression in the latter stages of the search can be seen more easily. In the graph legends, the names of the algorithms are referred to as follows:

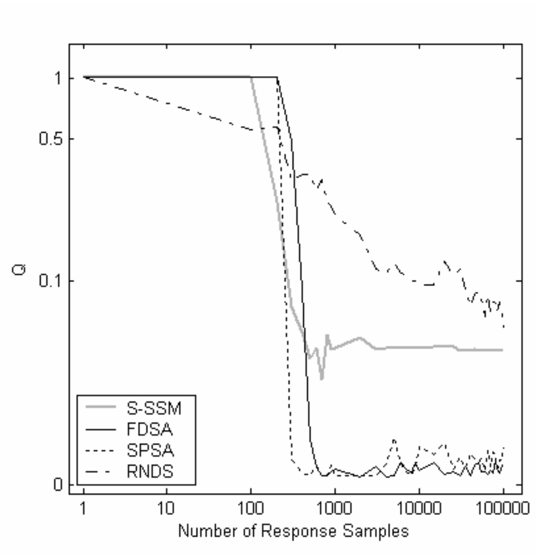
- RIN – MGPS with Rinott's procedure and no surrogates,
- SSM – MGPS with Sequential Selection with Memory procedure and no surrogates,
- SAS – MGPS with Screen-and-Select procedure and no surrogates,
- S-RIN – Surrogate assisted MGPS with Rinott's procedure,
- S-SSM – Surrogate assisted MGPS with Sequential Selection with Memory procedure,
- S-SAS – Surrogate assisted MGPS with Screen-and-Select procedure,
- FDSA – Finite-Difference Stochastic Approximation,
- SPSA – Simultaneous Perturbation Stochastic Approximation,
- RNDS – Random Search, and
- NM – Nelder-Mead simplex search.

For the continuous-variable problems, each of the MGPS-RS variants are plotted on the left of the page and on the right, all remaining algorithm implementations are plotted with S-SSM for a visual comparison with a MGPS-RS variant.

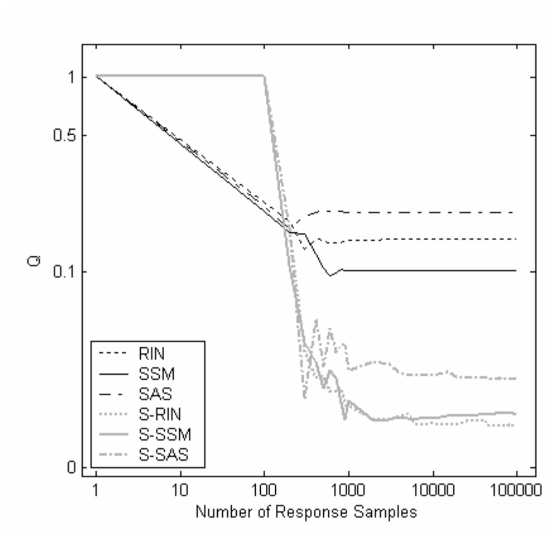
Test Problem 3

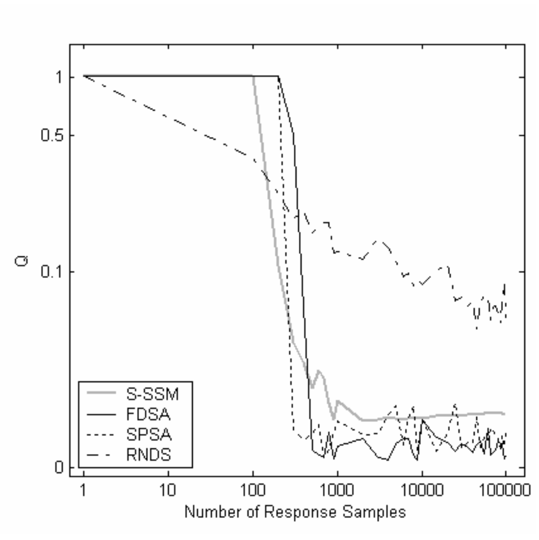
Noise Case 1





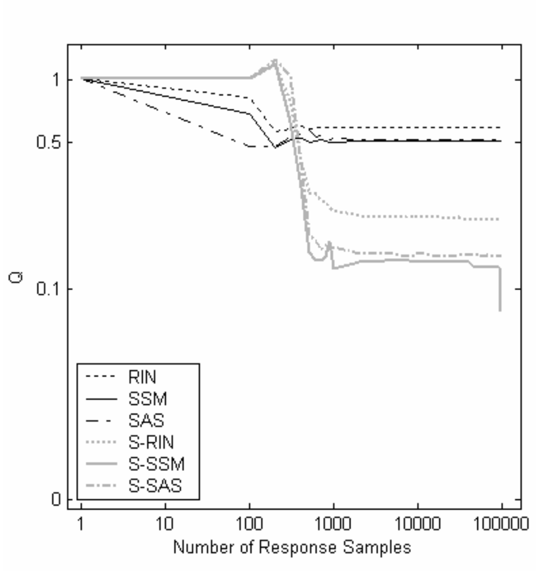
Noise Case 2

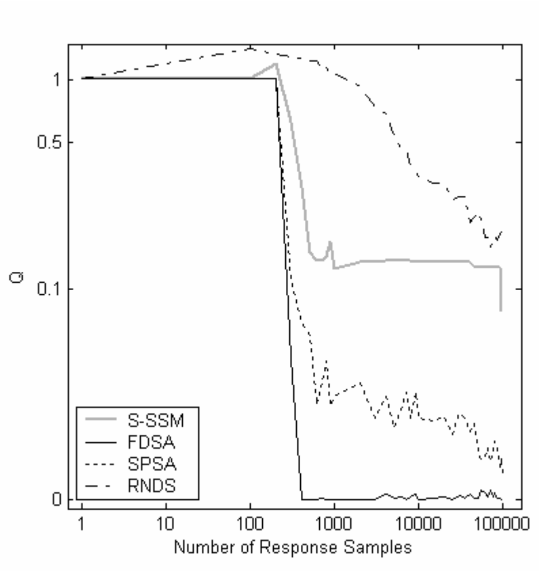




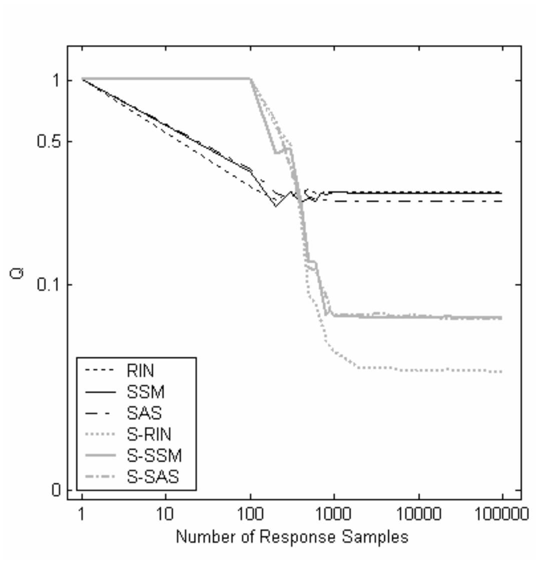
Test Problem 4

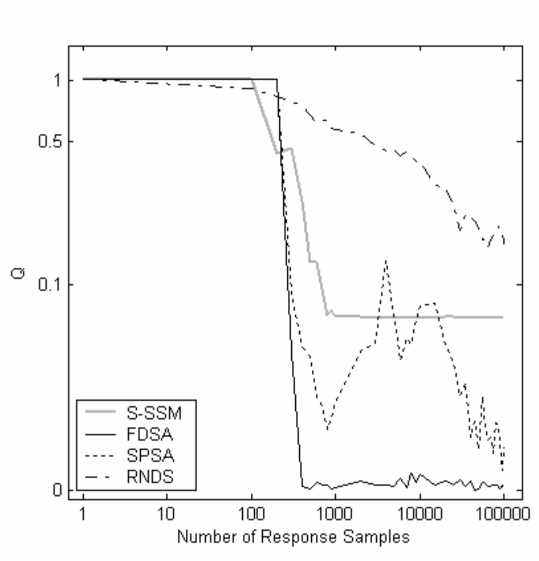
Noise Case 1





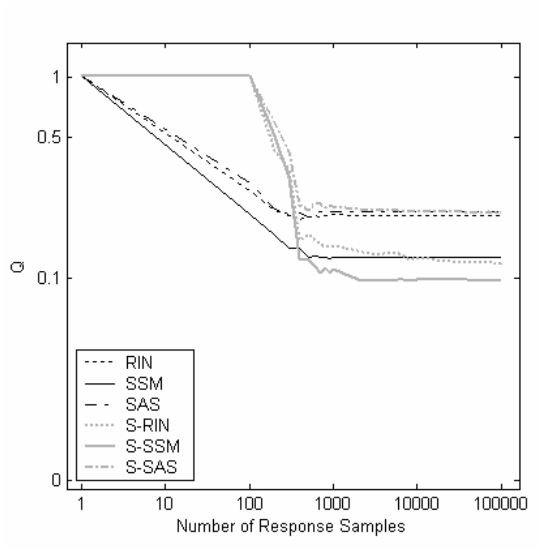
Noise Case 2

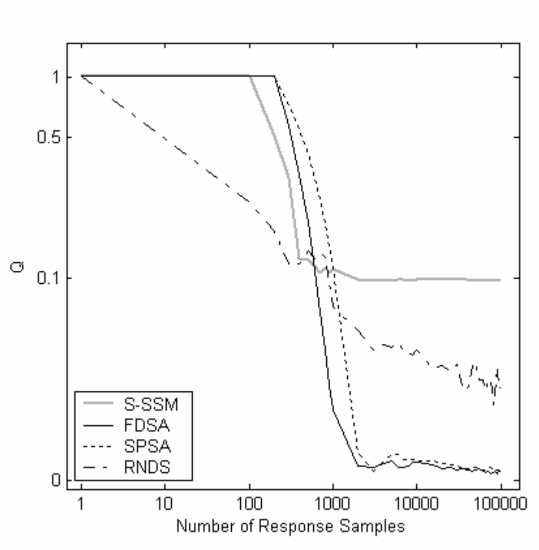




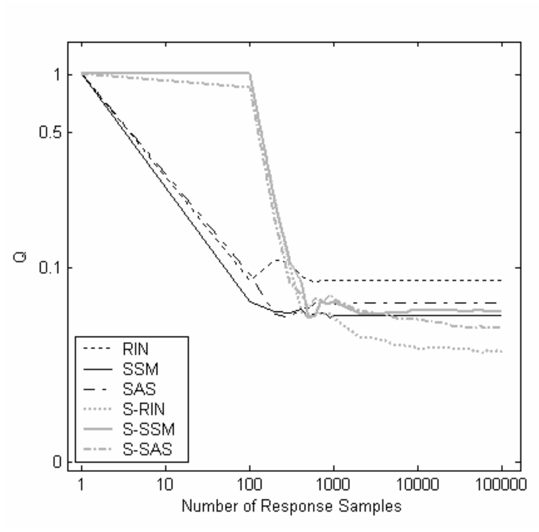
Test Problem 5

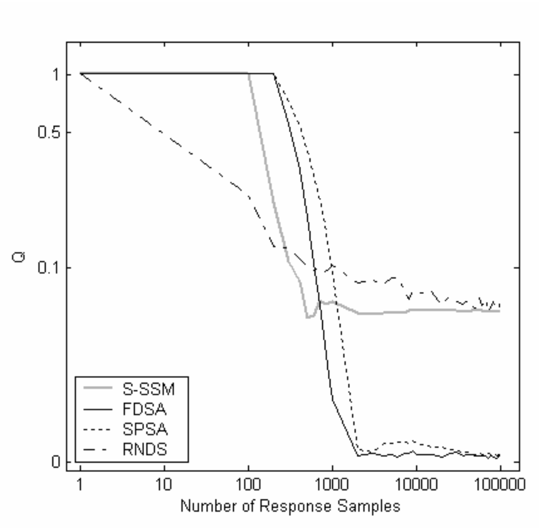
Noise Case 1





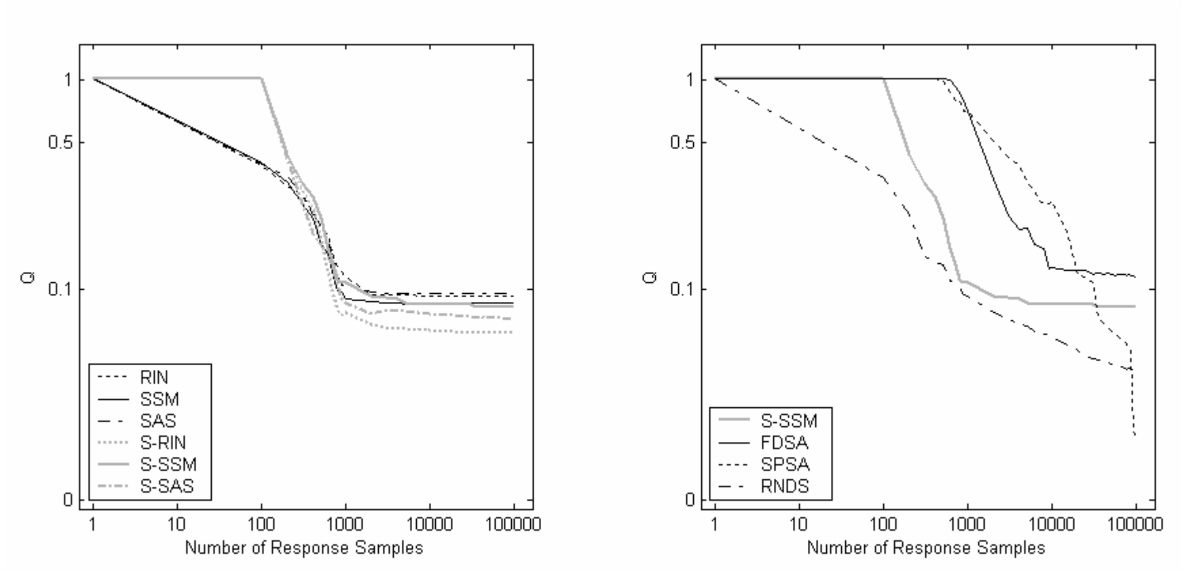
Noise Case 2



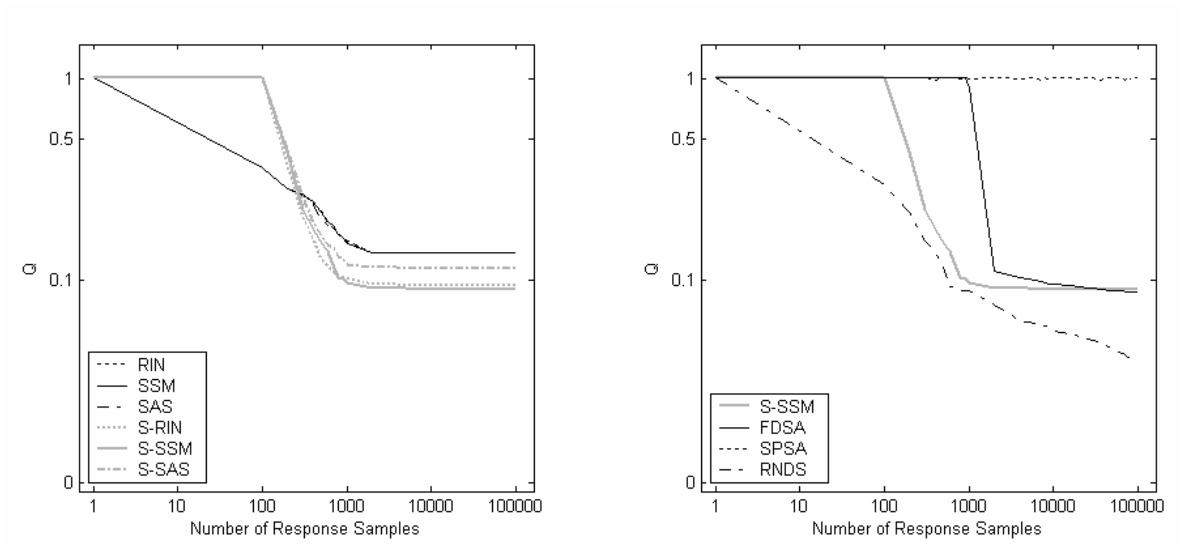


Test Problem 25

Noise Case 1

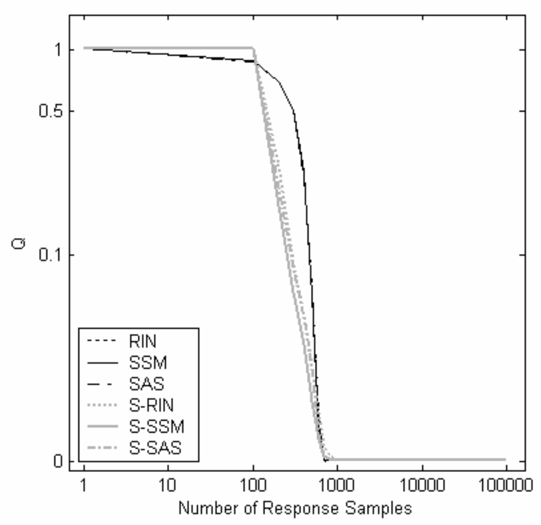


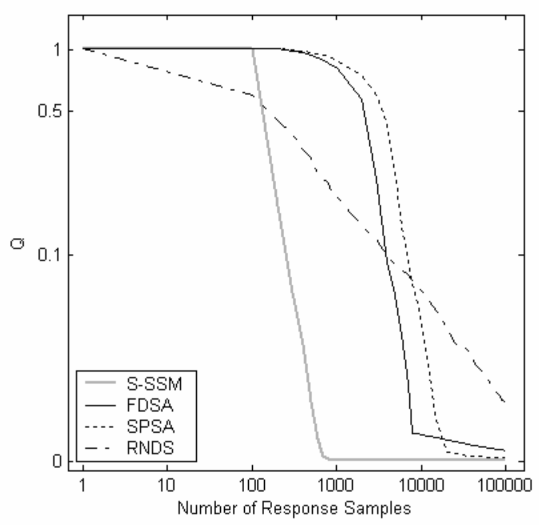
Noise Case 2



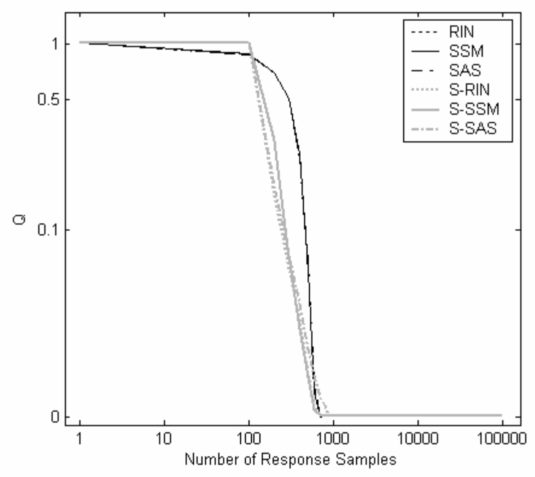
Test Problem 36

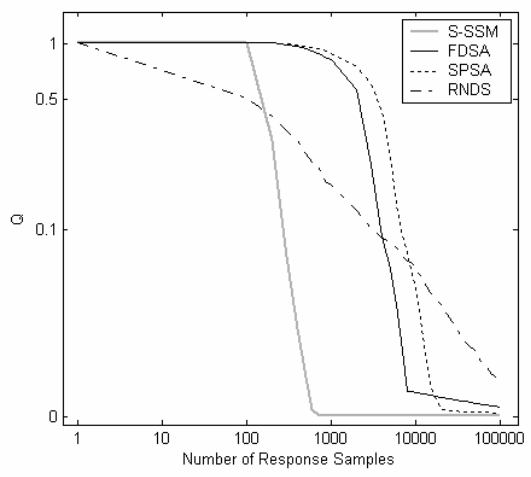
Noise Case 1





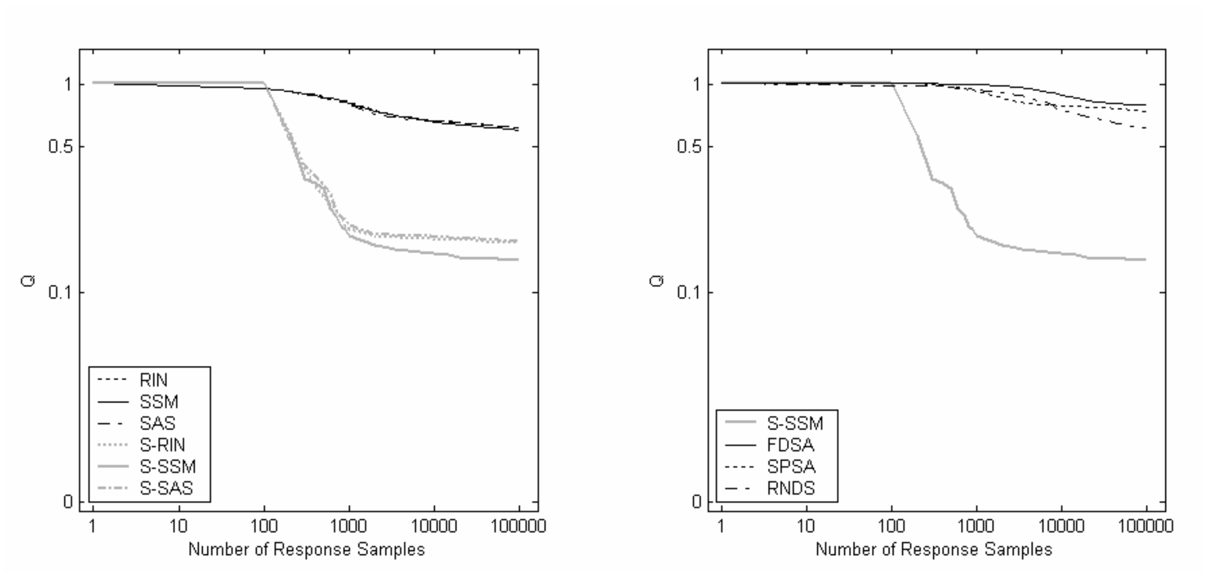
Noise Case 2



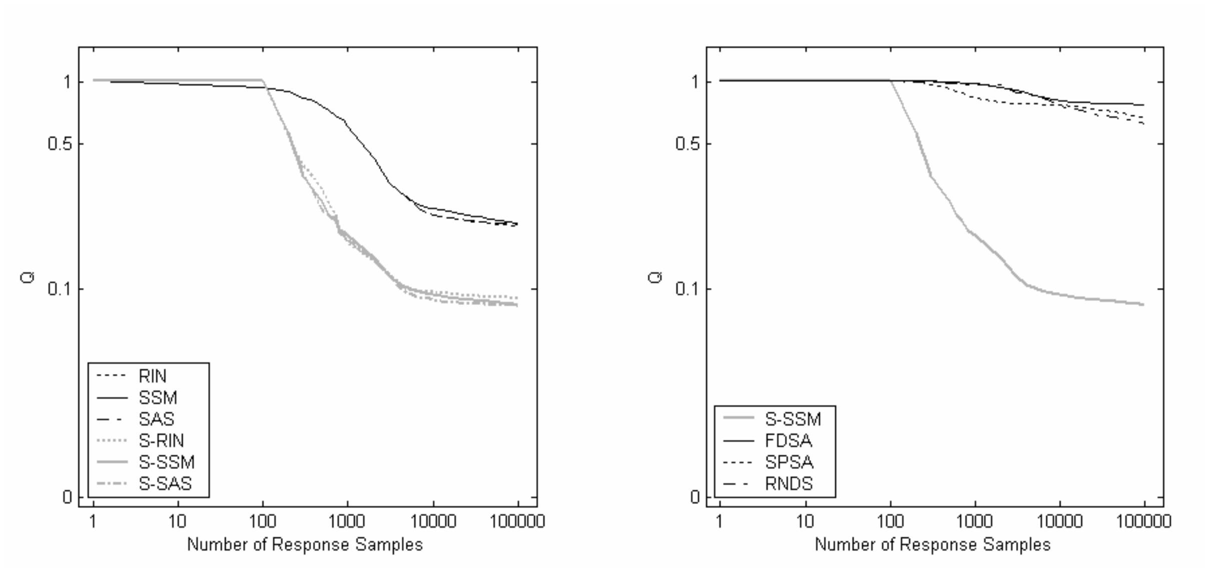


Test Problem 105

Noise Case 1

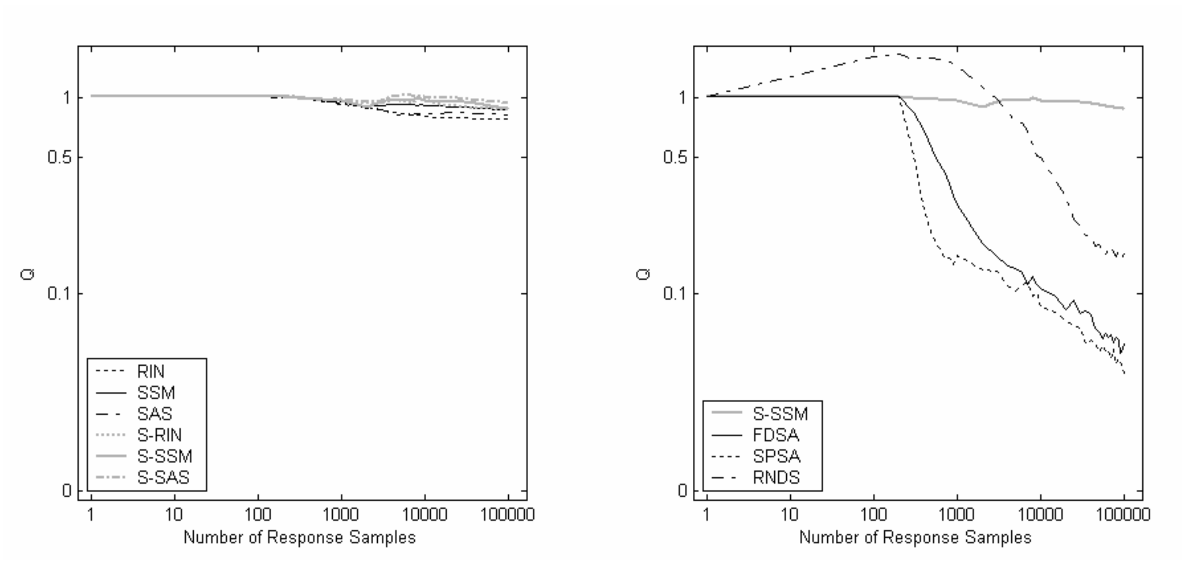


Noise Case 2

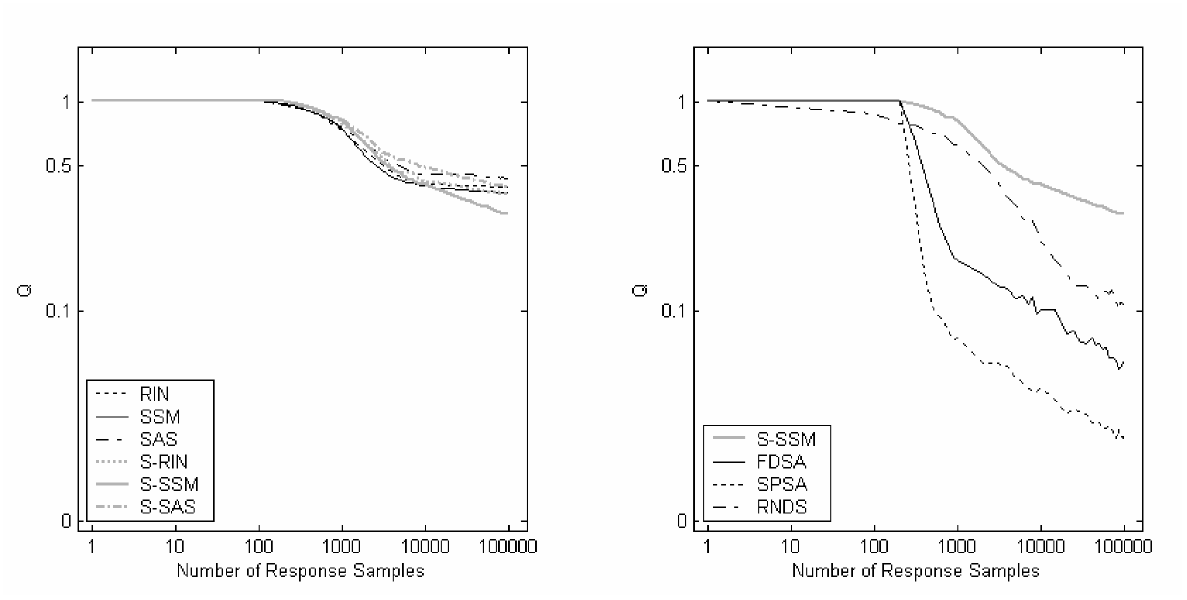


Test Problem 110

Noise Case 1

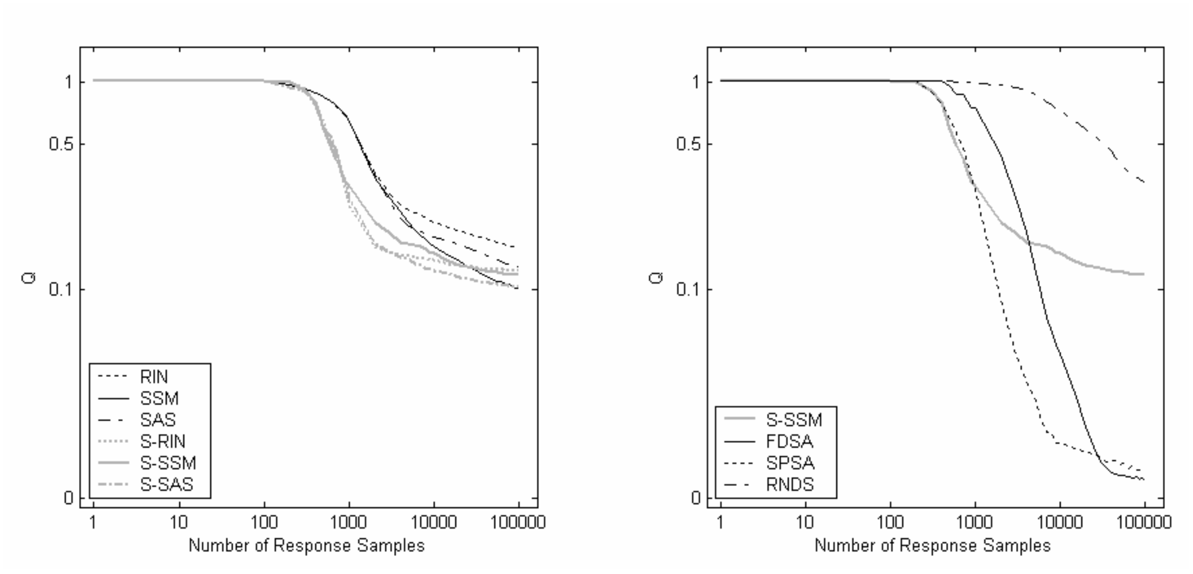


Noise Case 2

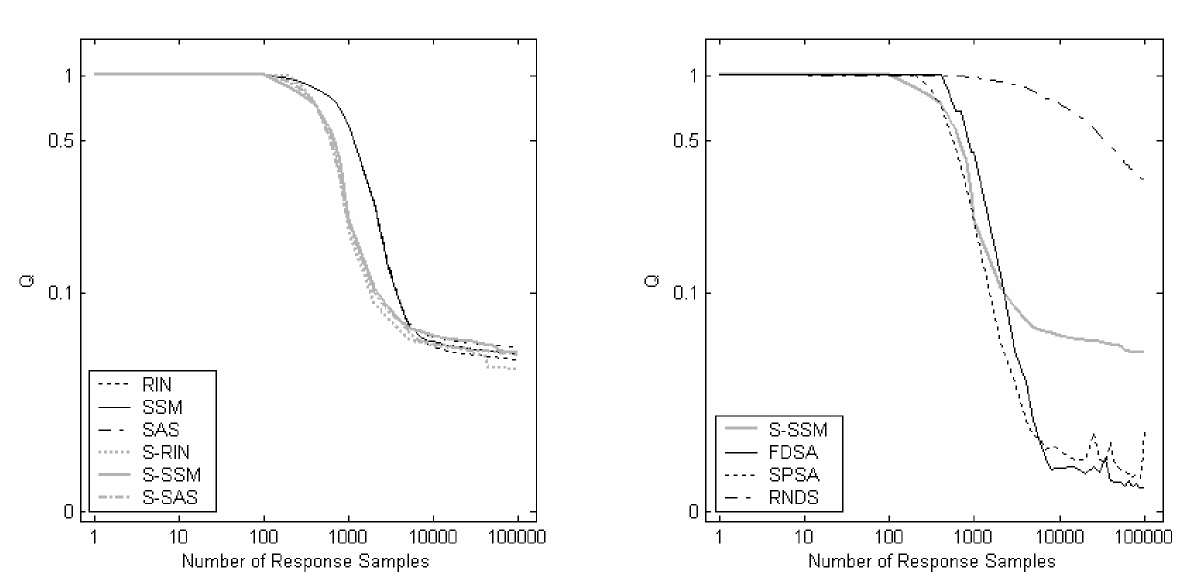


Test Problem 118

Noise Case 1

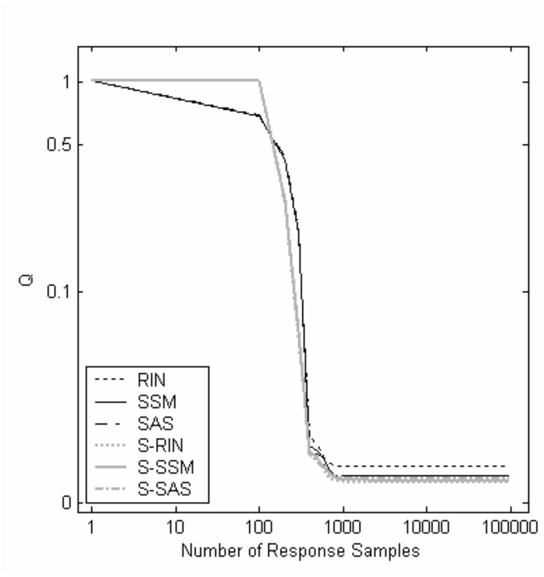


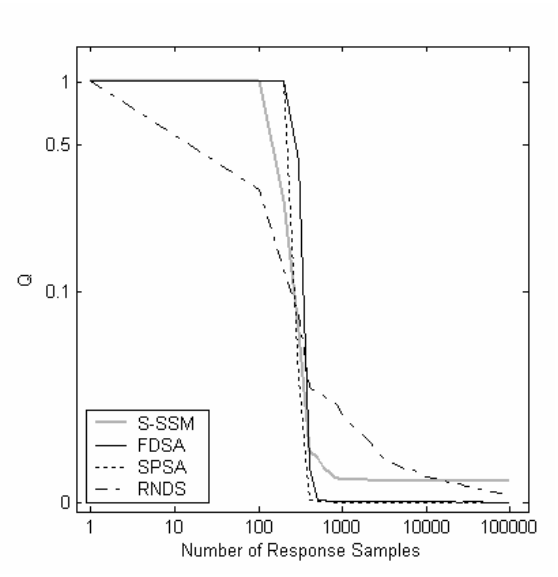
Noise Case 2



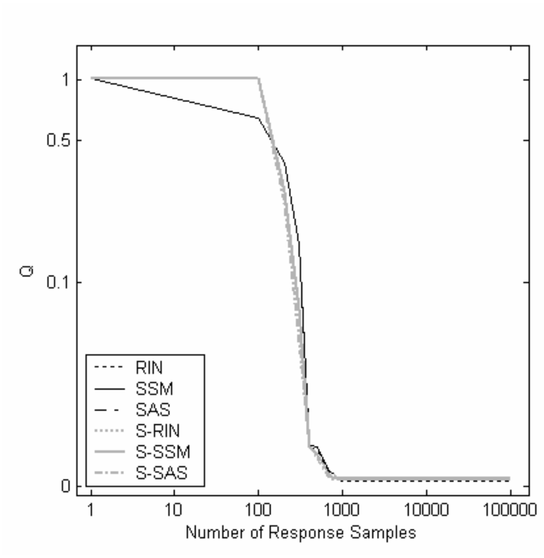
Test Problem 224

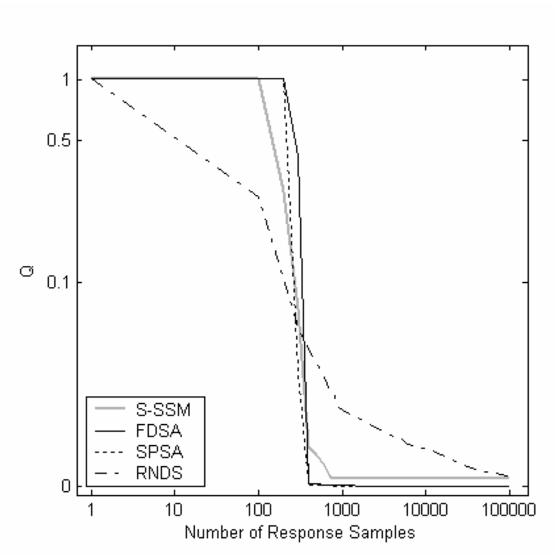
Noise Case 1





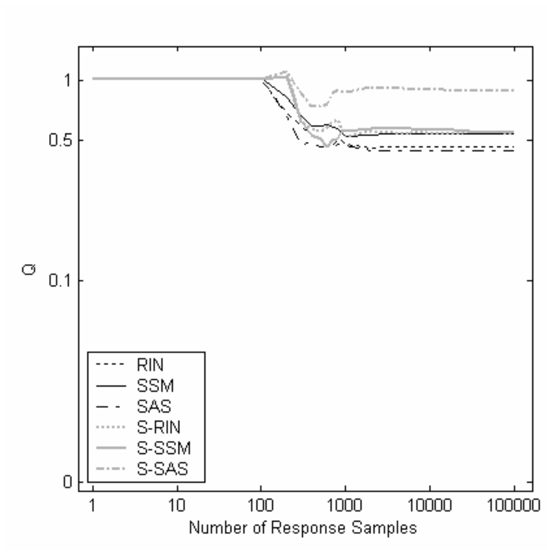
Noise Case 2

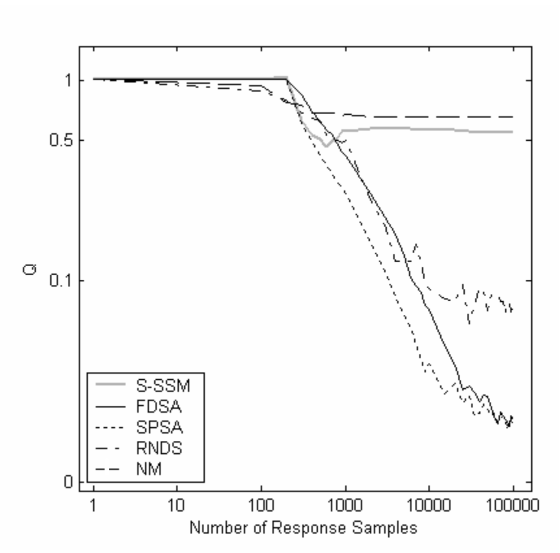




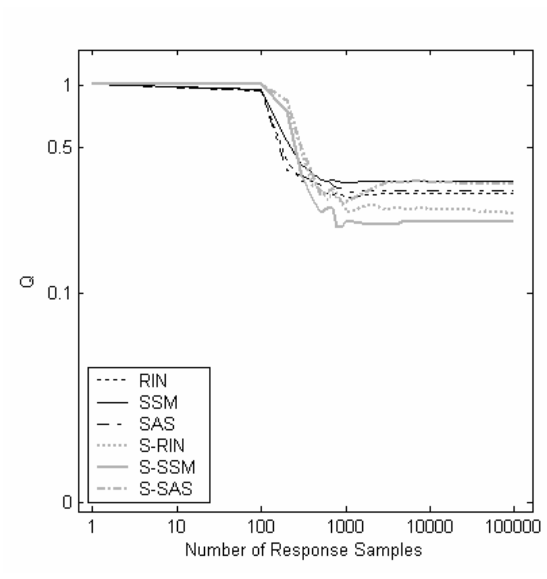
Test Problem 244

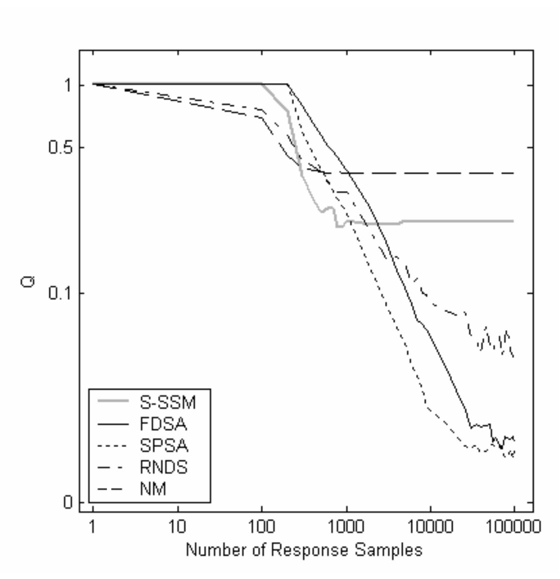
Noise Case 1





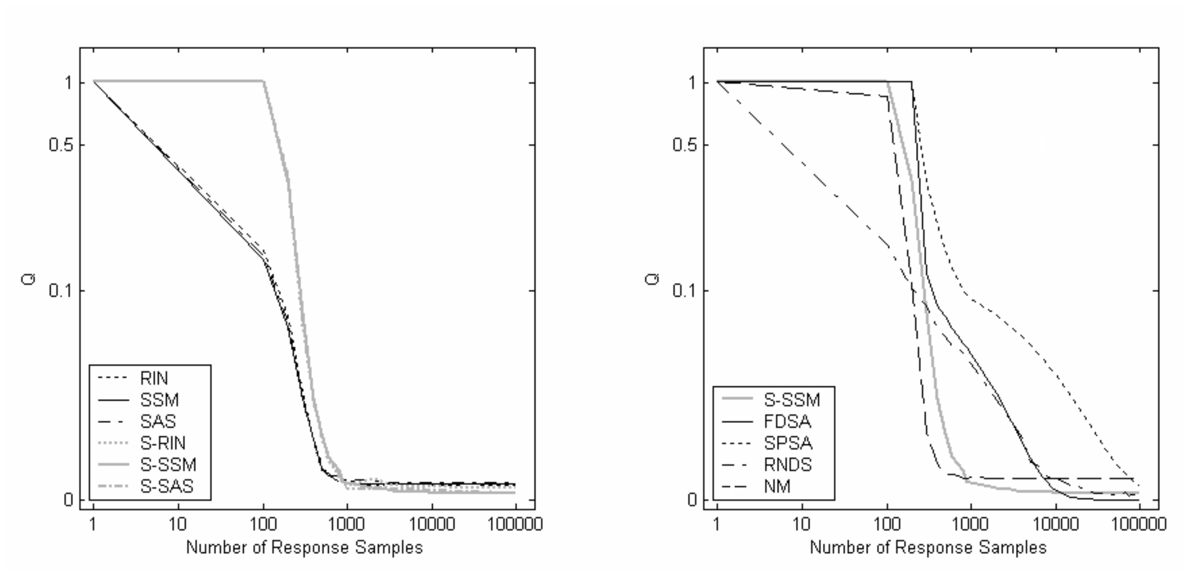
Noise Case 2



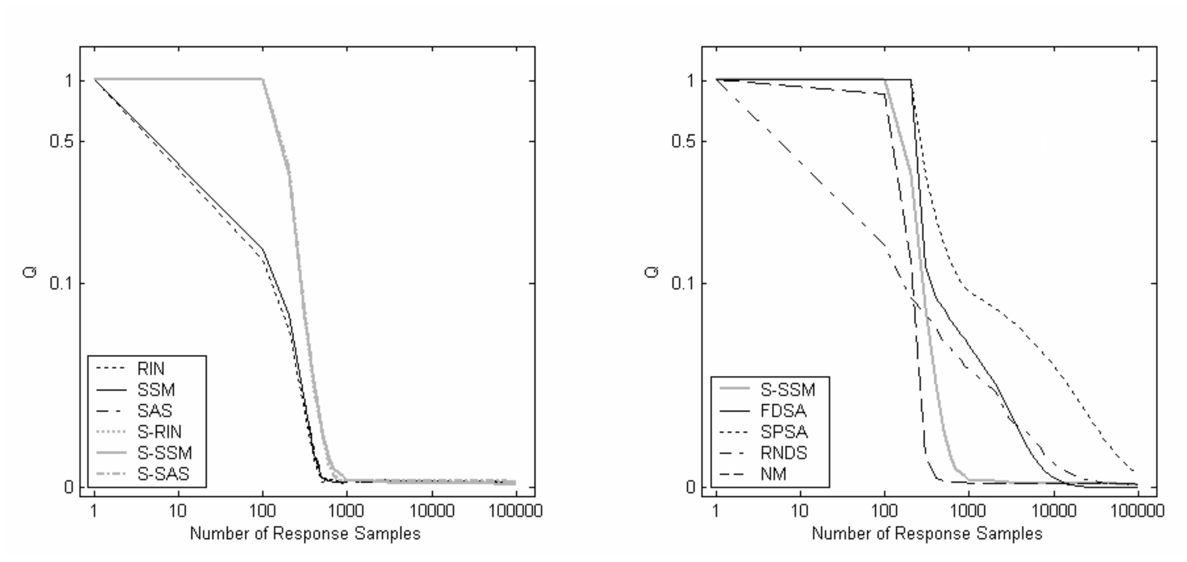


Test Problem 256

Noise Case 1

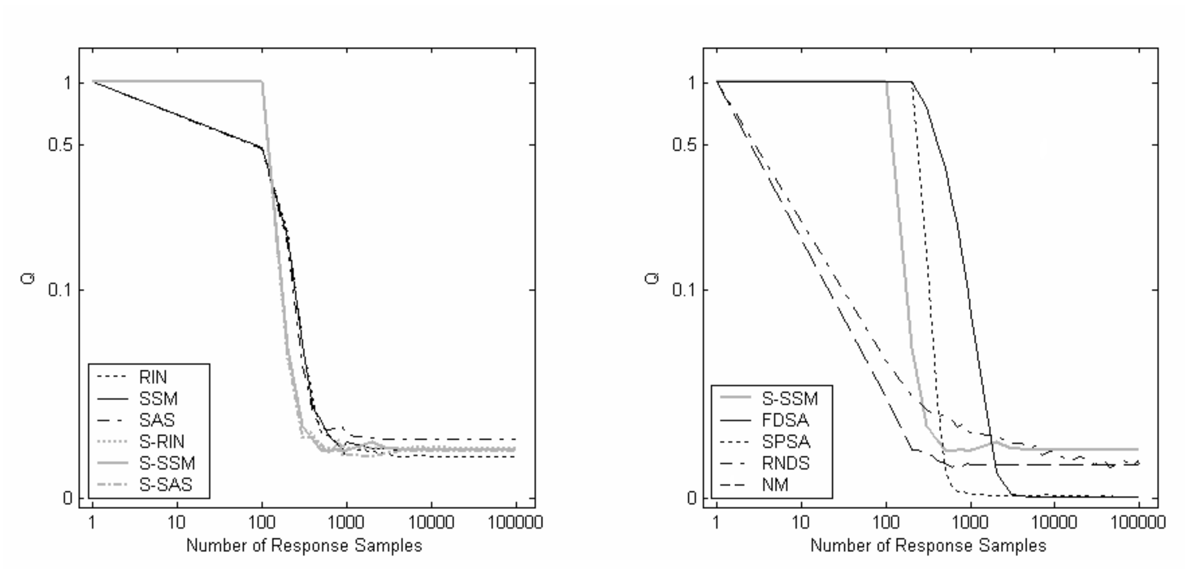


Noise Case 2

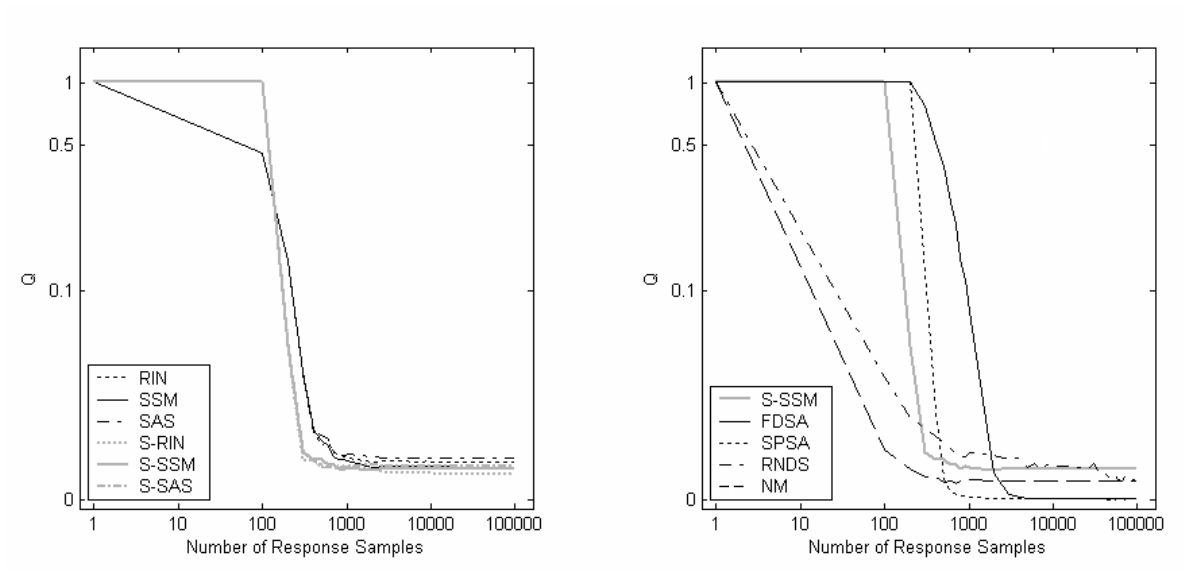


Test Problem 275

Noise Case 1

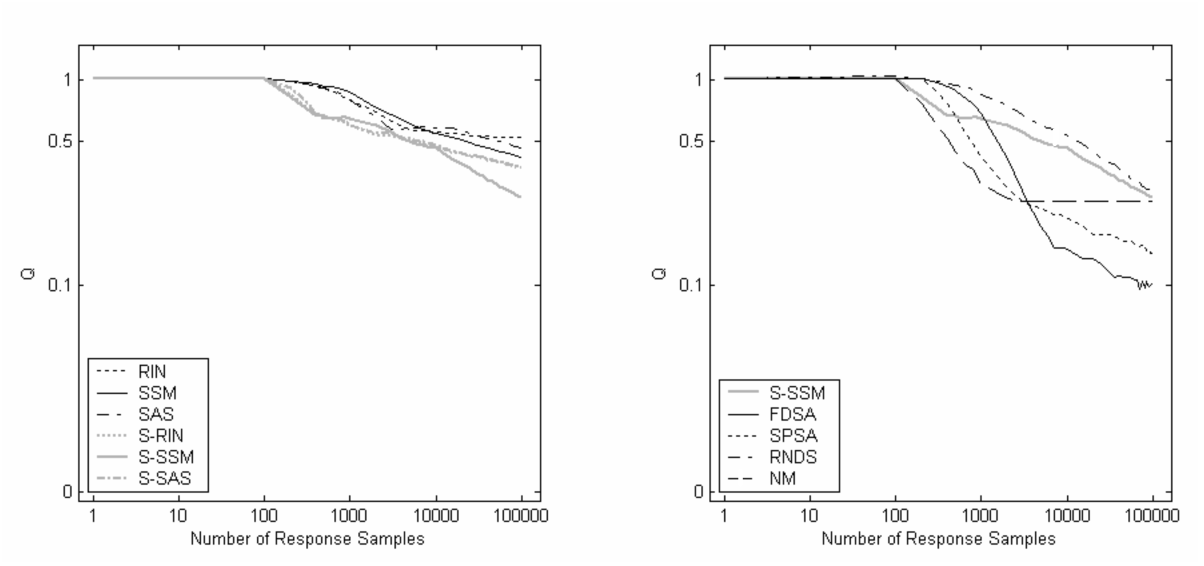


Noise Case 2

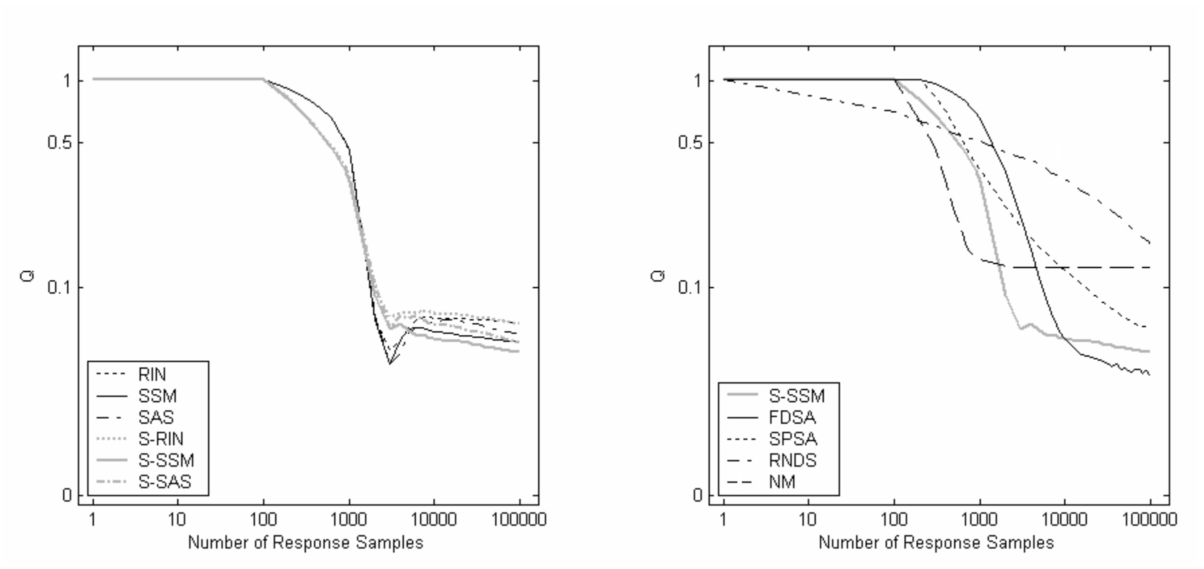


Test Problem 281

Noise Case 1

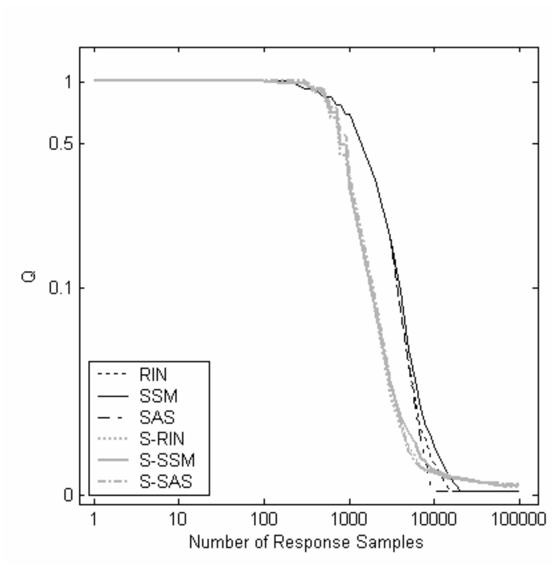


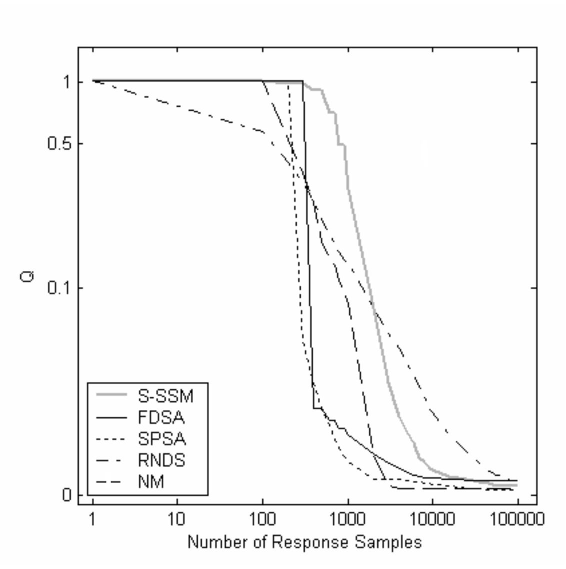
Noise Case 2



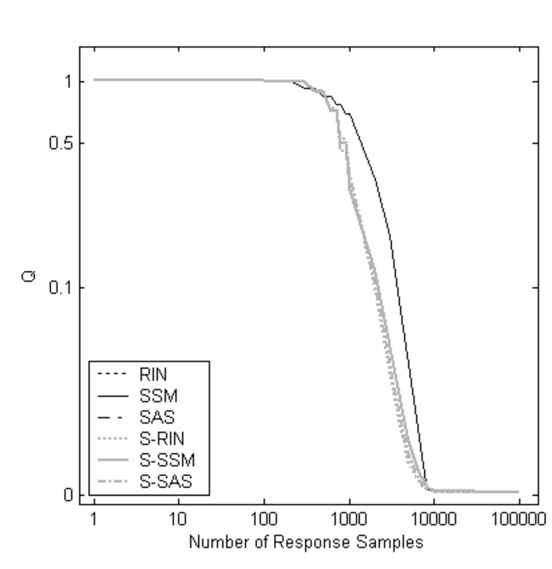
Test Problem 287

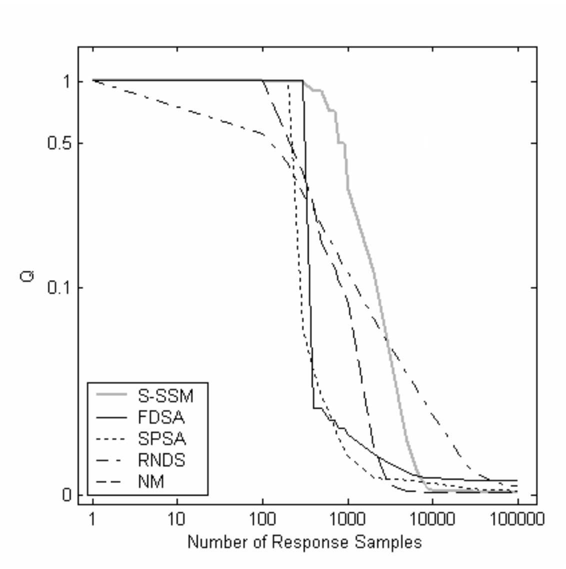
Noise Case 1





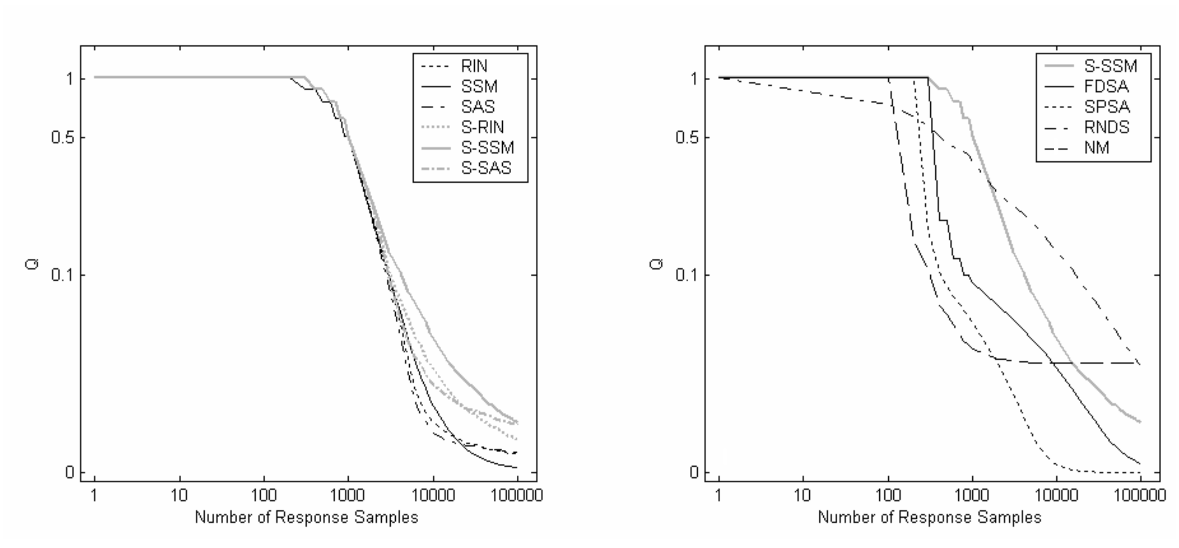
Noise Case 2



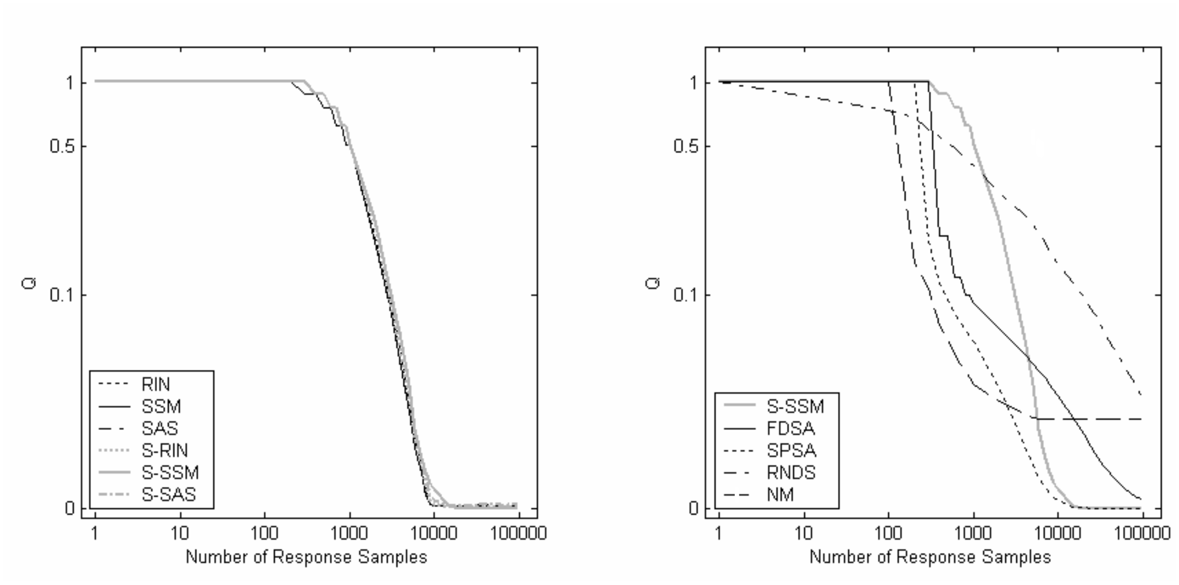


Test Problem 288

Noise Case 1

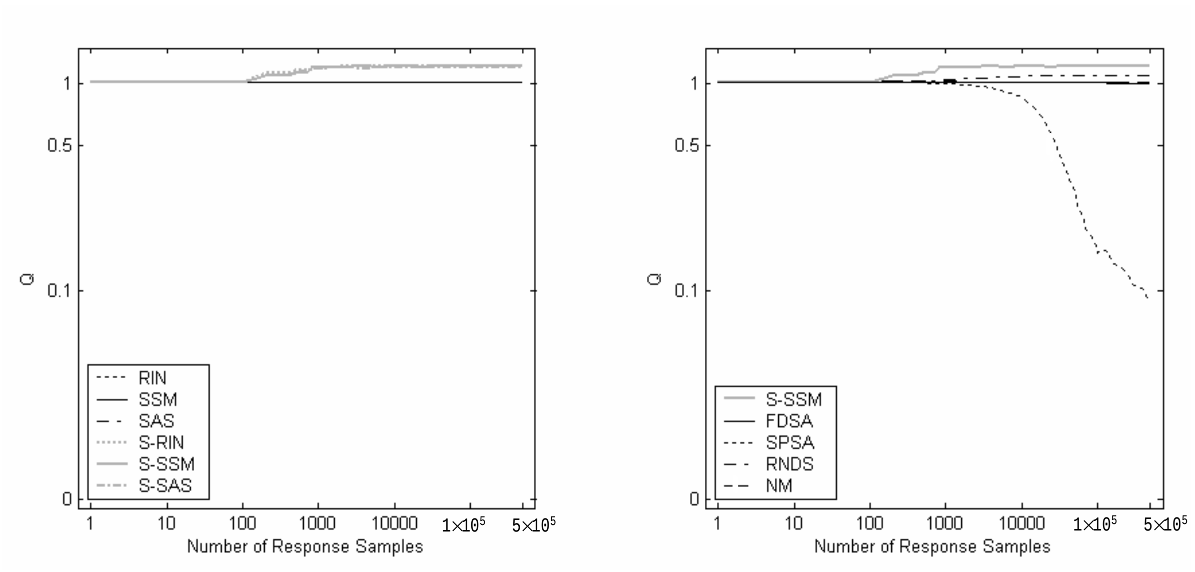


Noise Case 2

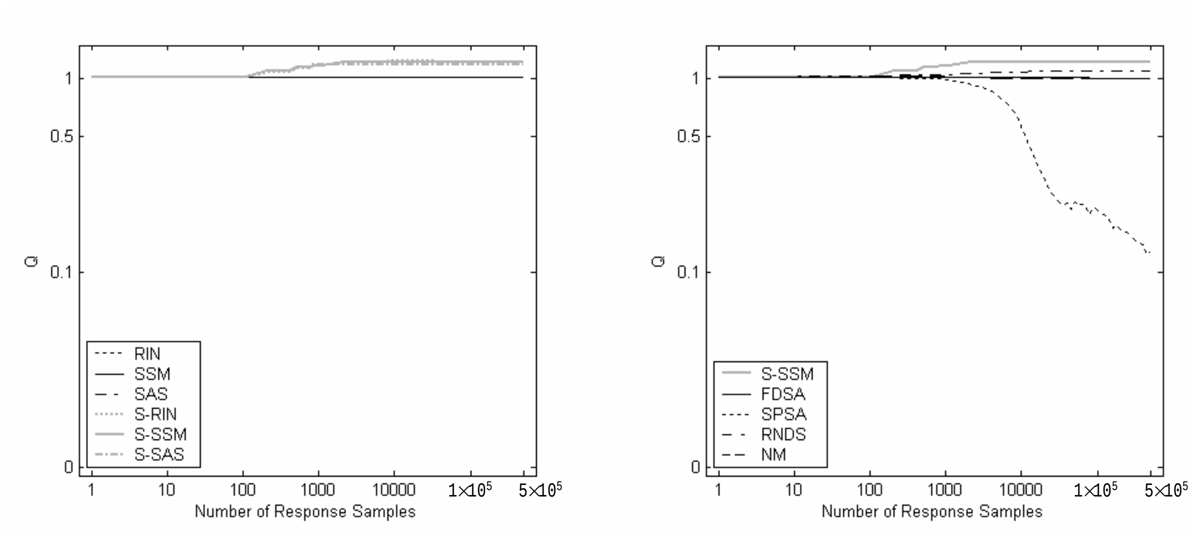


Test Problem 289

Noise Case 1

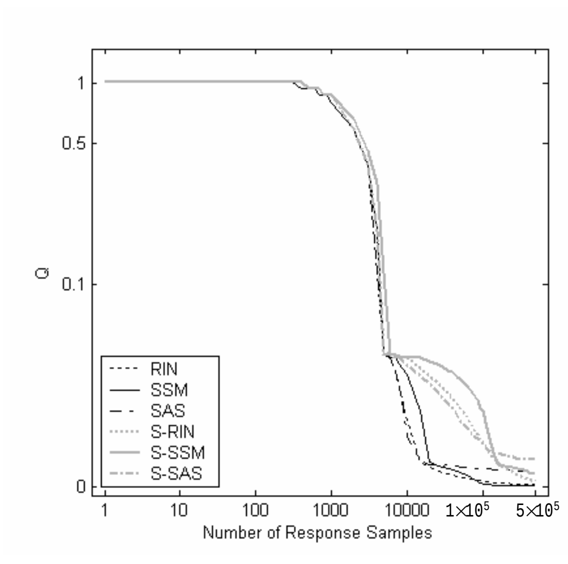


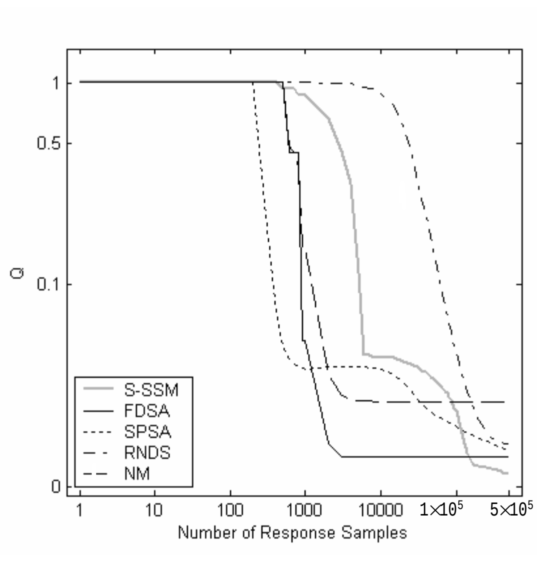
Noise Case 2



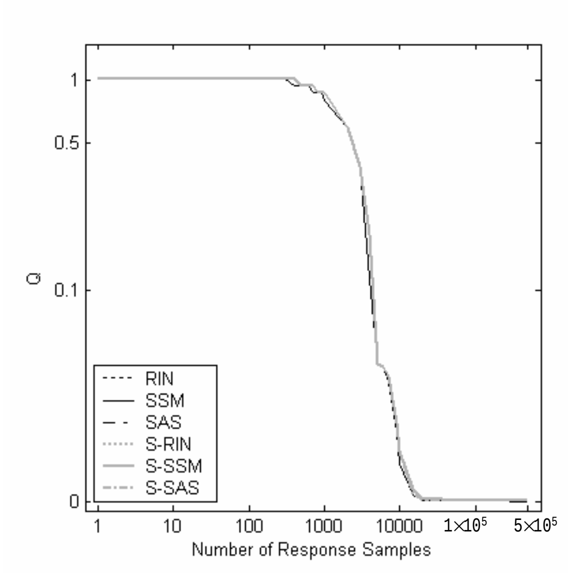
Test Problem 297

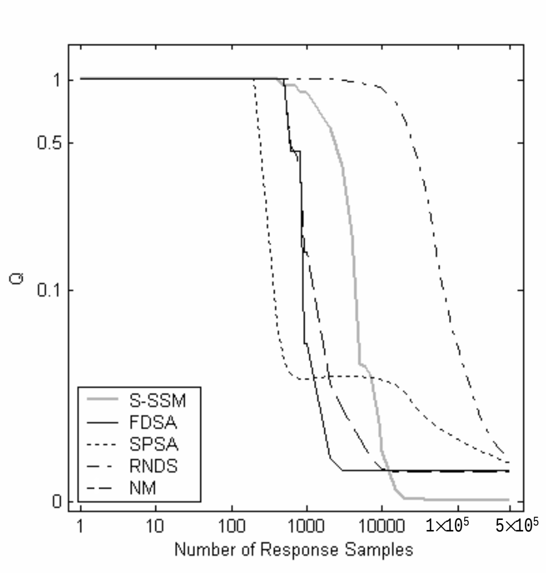
Noise Case 1





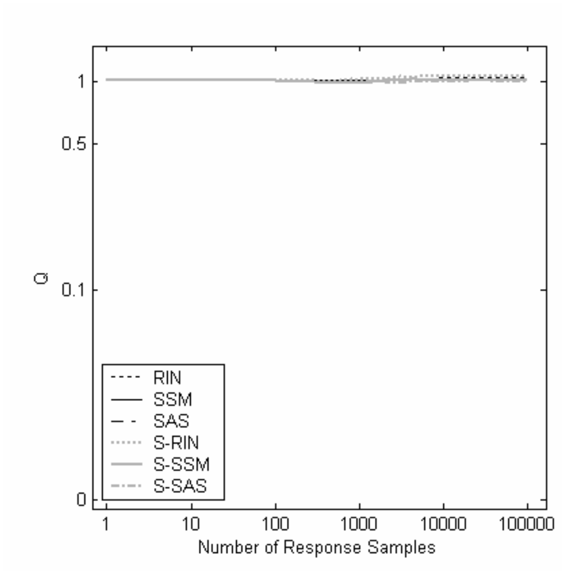
Noise Case 2

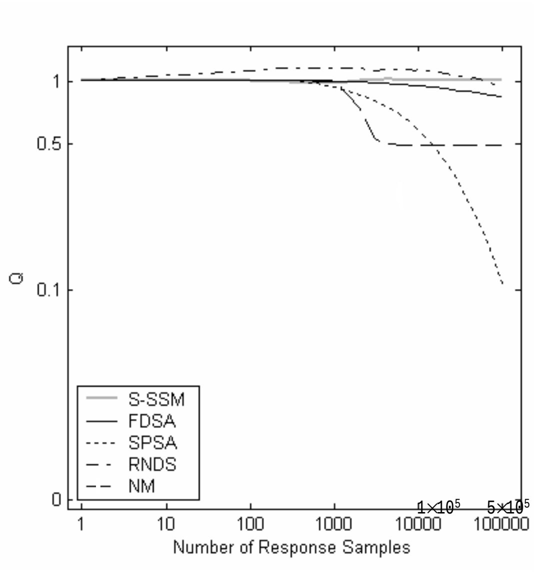




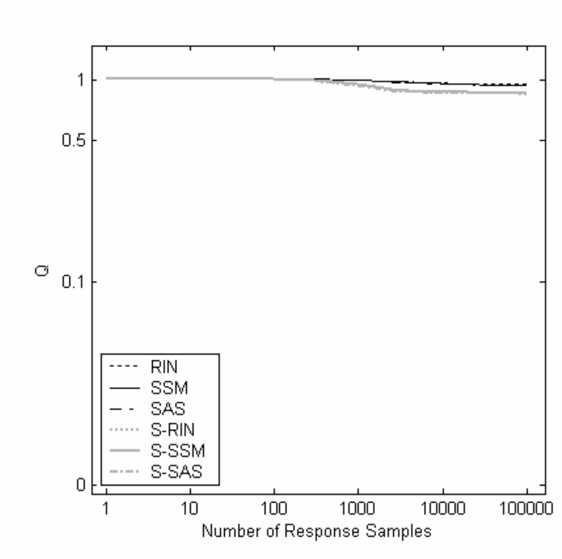
Test Problem 300

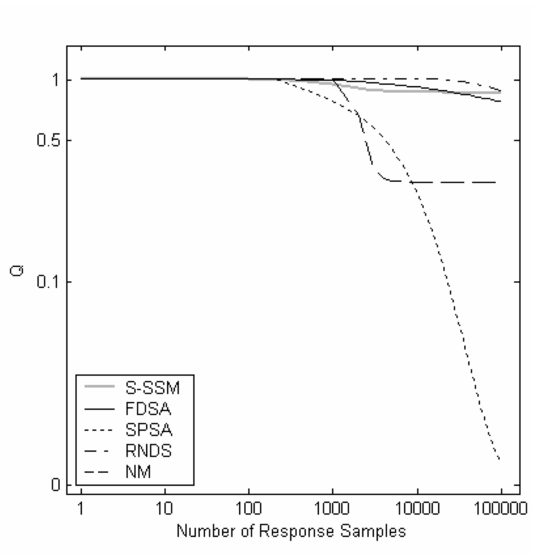
Noise Case 1





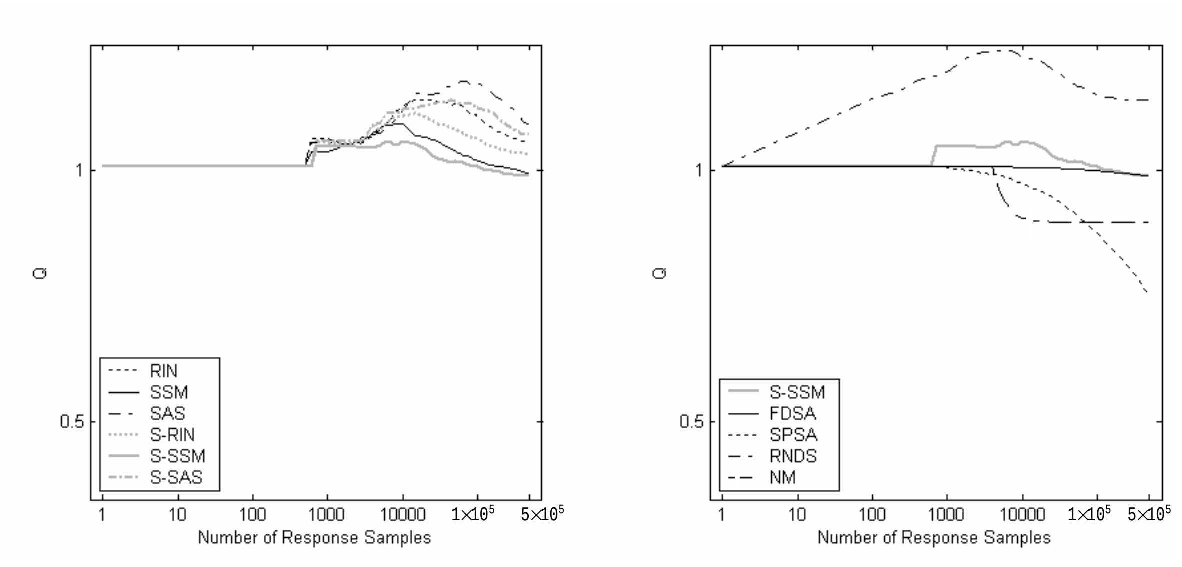
Noise Case 2



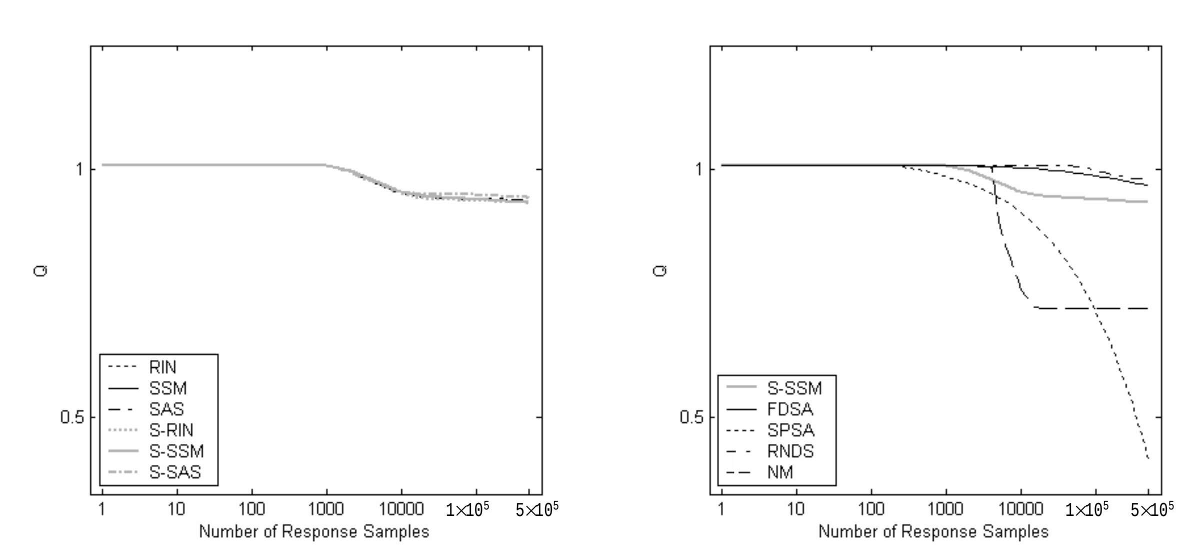


Test Problem 301

Noise Case 1

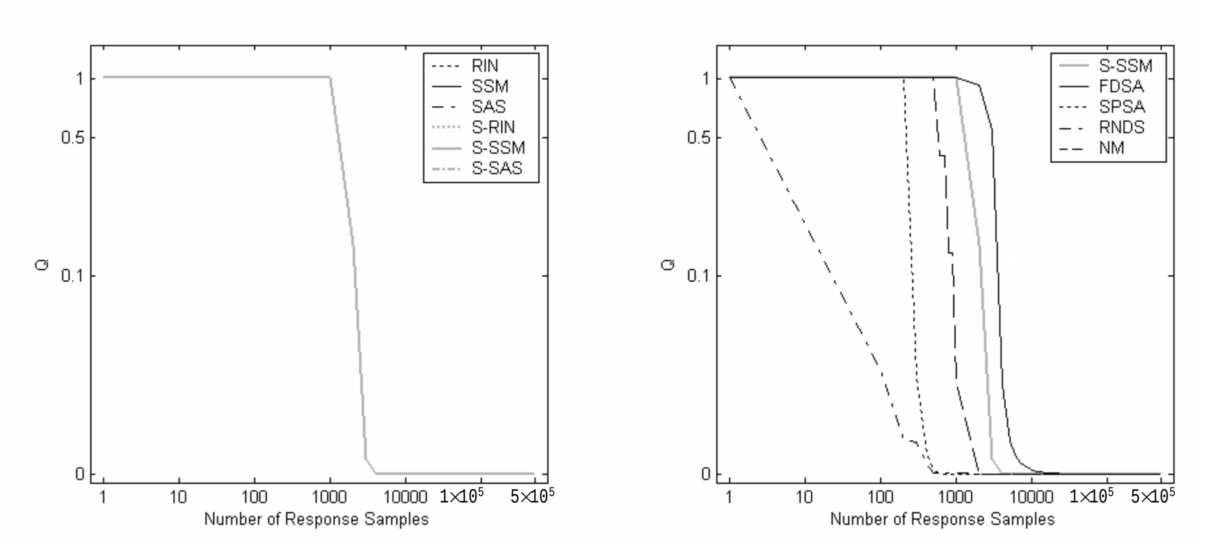


Noise Case 2

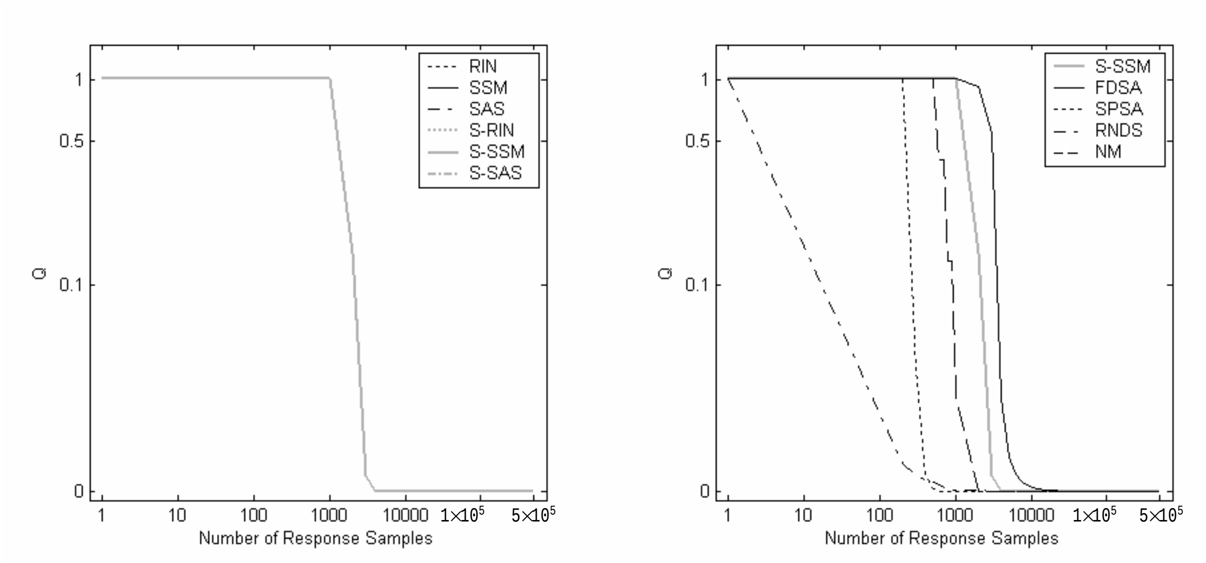


Test Problem 305

Noise Case 1

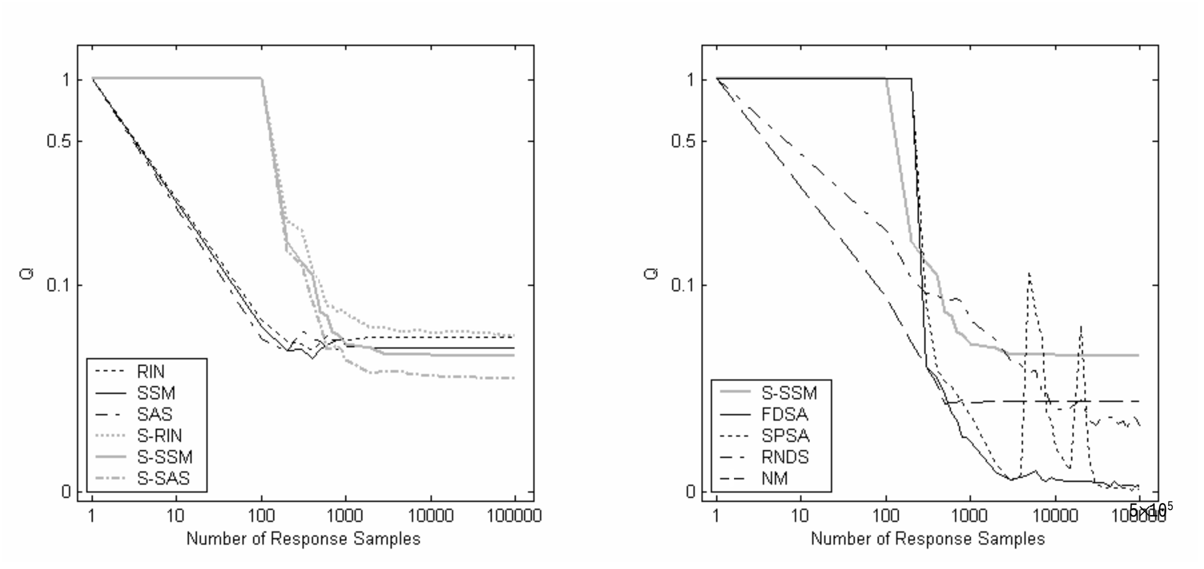


Noise Case 2

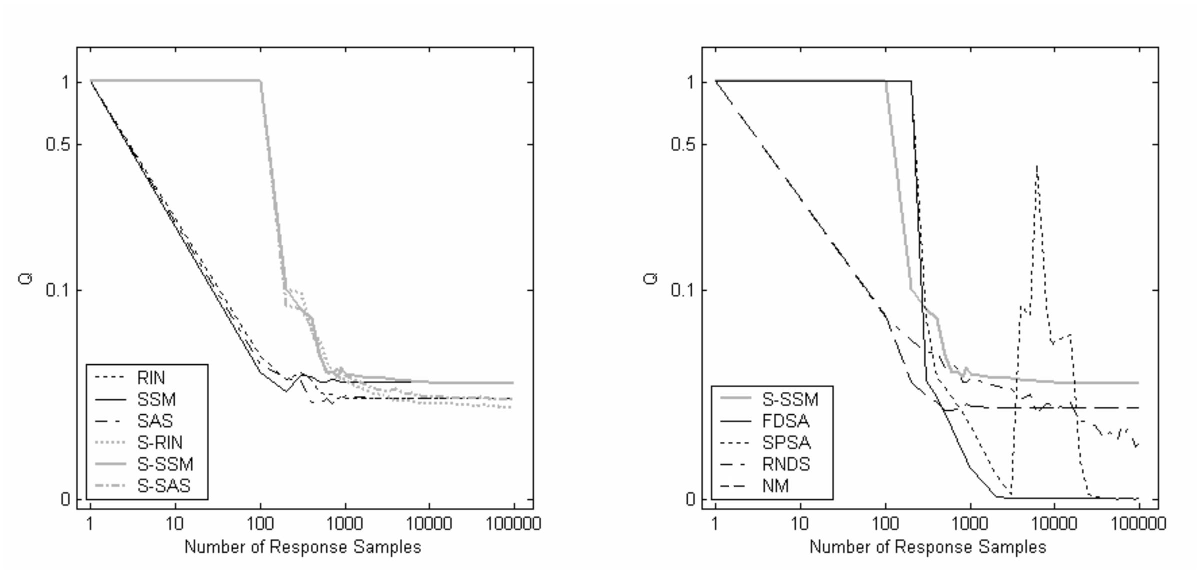


Test Problem 314

Noise Case 1

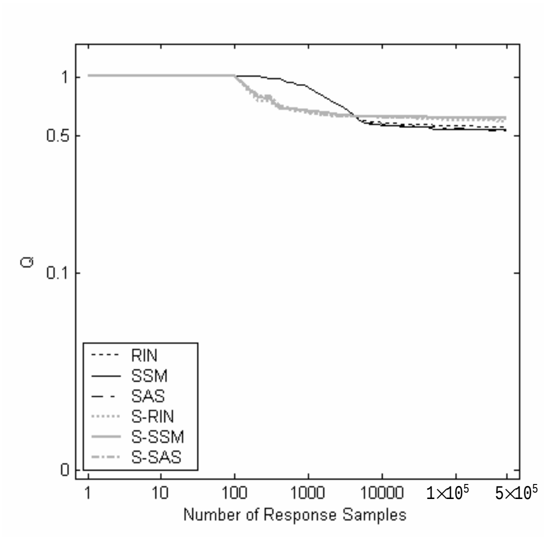


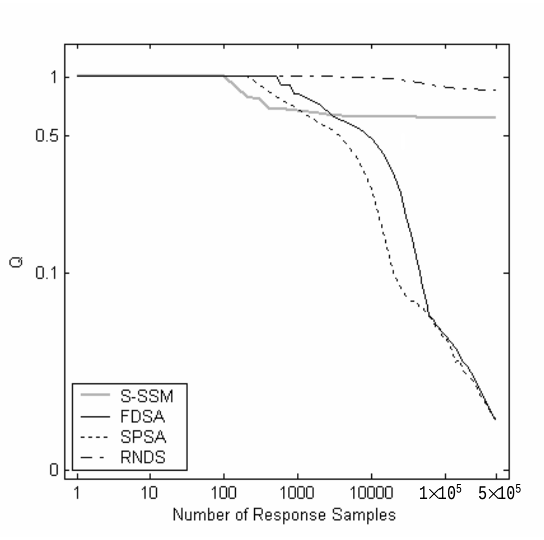
Noise Case 2



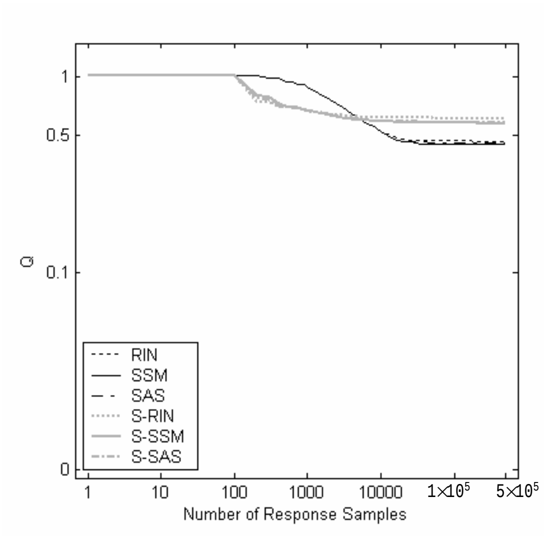
Test Problem 392

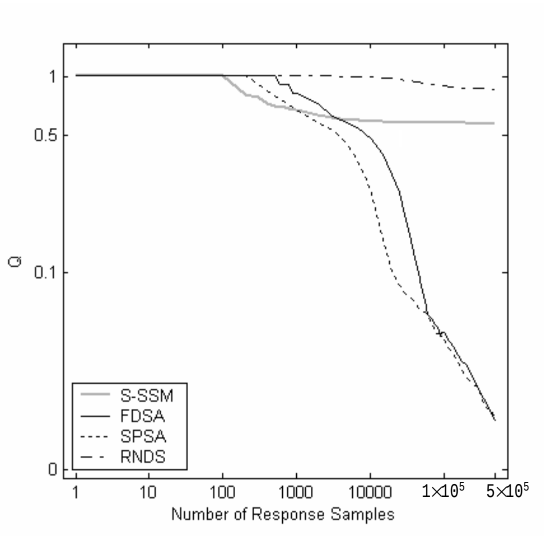
Noise Case 1





Noise Case 2

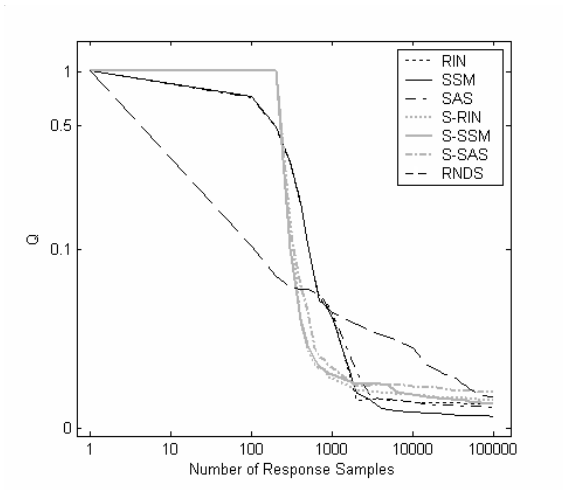


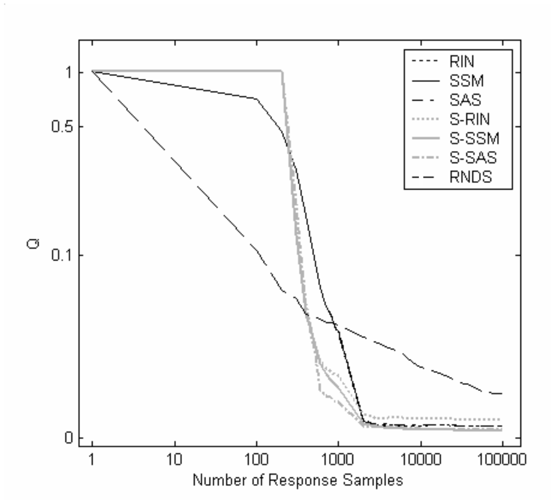


Test Problem MVP1

Noise Case 1

Noise Case 2

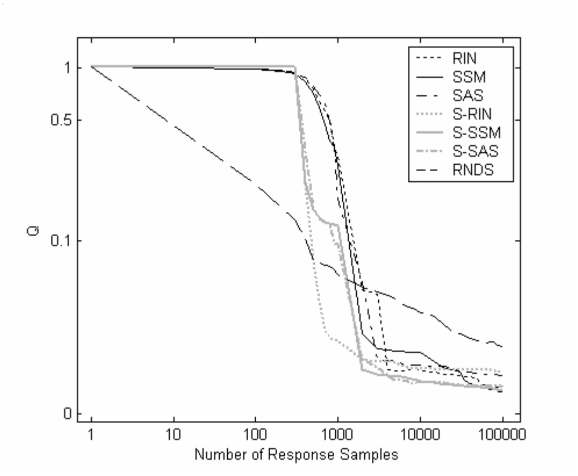


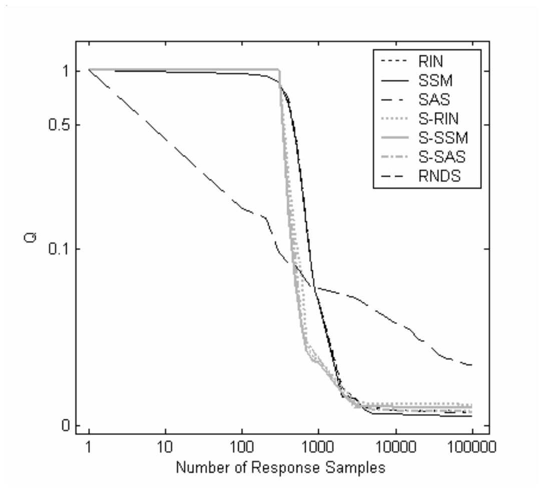


Test Problem MVP2

Noise Case 1

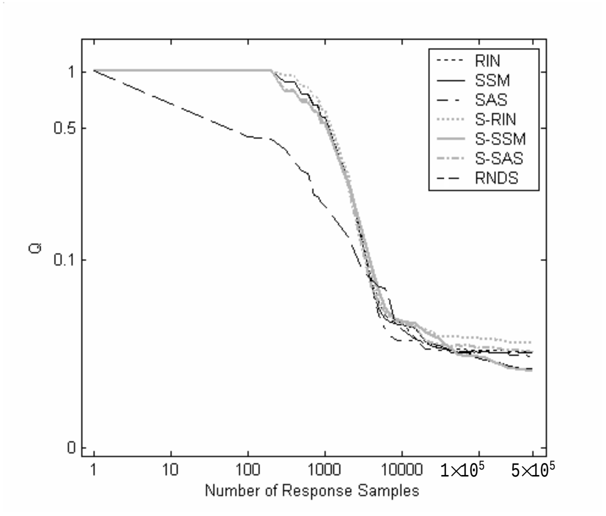
Noise Case 2



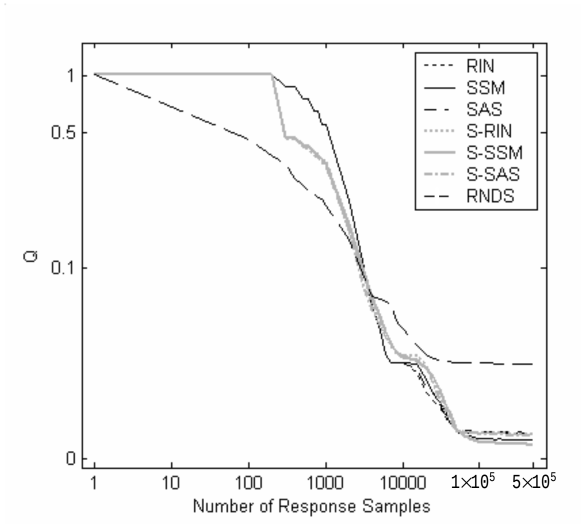


Test Problem MVP3

Noise Case 1

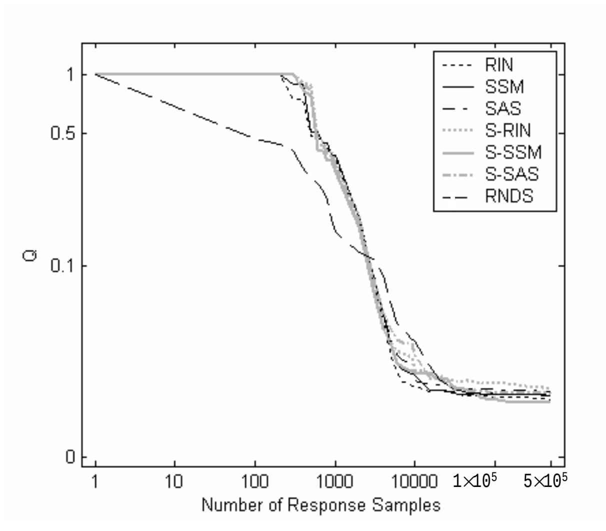


Noise Case 2

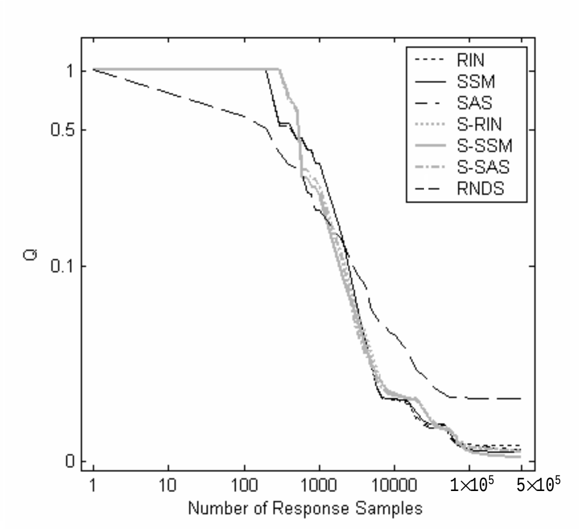


Test Problem MVP4

Noise Case 1



Noise Case 2



B.2 Statistical Analysis Data Summary

Data necessary to support the ANOVA of Section 5.5.1 is provided in Table B.1 for each of the performance measures Q and P. The transformation function used for experiment outcome $Q_{ijk\ell}$ and $P_{ijk\ell}$, for $1 \leq i \leq 3$ (index on R&S procedure), $1 \leq j \leq 2$ (index on the use of surrogates), $1 \leq k \leq 2$ (index on noise case), and $1 \leq \ell \leq 30$ (index on replication) is shown in the $T(Q_{ijk\ell})$ and $T(P_{ijk\ell})$ column, respectively. Also shown is the test statistic W for the Shapiro-Wilk test for nonnormality of the studentized residuals and the p-value associated with that test. The closer W is to unity, the more likely the data will be accepted as normal. The null hypothesis is that the data are normal, so a p-value greater than .05 would indicate that the null hypothesis is not rejected at a .05 significance level. On the pages following Table B.3 a series of charts is shown for each test problem that plots the studentized residuals versus their predicted values as a visual test of the constant variance assumption, and the normal probability plot of the studentized residuals as a visual test of the normality assumption.

The results of the nonparametric tests are presented in Tables B.2 and B.3 for performance measures Q and P, respectively. In the tables, p-values are displayed that test for differences in the distributions as a result of the R&S procedure (RS), using surrogates (SRCH), and noise. The column headings are defined as follows:

- WIL – Wilcoxon rank-sum procedure,
- KW – Kruskal-Wallis procedure,
- MED – two-sample median procedure,
- BM – Brown-Mood procedure, and
- VDW – van der Waerden procedure.

Both WIL and KW test the null hypothesis that the independent samples represent populations with the same median value. The MED, BM, and VDW procedures test the null hypothesis that the independent samples derive from the same distribution. In all cases, a p-value greater than 0.05 indicates that the null hypothesis is not rejected at the .05 significance level. P-values that are enclosed by a rectangle signify tests that disagree with the results of the ANOVA and multiple comparison procedures of Section 5.5.1. For example, in test problem 36, the ANOVA procedure specified the effect of the R&S procedure to be significant at the .05 level for performance measure Q, but each of the three nonparametric tests fail to reject their null hypotheses at this significance level.

Table B.1. **Transformation functions and Shapiro-Wilk Nonnormality Test Results.**

Test	Performance Measure Q			Performance Measure P		
Problem	$T(Q_{ijk\ell})$	W	p-value	$T(P_{ijk\ell})$	W	p-value
3	$\log(Q)$	.993	.199	$P^{\frac{1}{2}}$	.858	.000
4	$\log(Q+1)$	.806	.000	$(P+1)^{-\frac{1}{2}}$	.854	.000
5	$\log(Q)$	.990	.015	$\log(P)$	.983	.000
25	Q^2	.986	.002	P	.982	.000
36	$\log(Q)$	.963	.000	$\log(P)$	.968	.000
105	Q	.925	.000	P	.903	.000
110	Q	.990	.015	P	.993	.078
118	$\log(Q)$	.961	.000	$\log(P)$	.946	.000
224	$\log(Q)$	.980	.000	$P^{\frac{1}{2}}$	.992	.059
244	$\log(Q)$	.967	.000	P	.990	.016
256	$\log(Q)$	.977	.000	$P^{\frac{1}{2}}$	.988	.004
275	$\log(Q)$	.966	.000	$P^{\frac{1}{2}}$	.998	.879
281	$(Q+1)^{-1}$	.951	.000	$P^{\frac{1}{2}}$	.964	.000
287	$Q^{-\frac{1}{2}}$	.862	.000	$P^{-\frac{1}{2}}$	.901	.000
288	$\log(Q)$	.980	.000	$\log(P)$	.982	.000
289	$Q^{\frac{1}{2}}$	.886	.000	P^{-1}	.895	.000
297	$\log(Q)$	.982	.000	$\log(P)$	.904	.000
300	$\log(Q)$	.931	.000	$\log(P)$	.825	.000
301	$\log(Q)$	.965	.000	P	.957	.000
305	$\log(Q)$	.788	.000	P^{-1}	.772	.000
314	$\log(Q)$	.939	.000	$P^{\frac{1}{2}}$	.990	.017
392	Q	.889	.000	P	.931	.000
MVP1	$\log(Q)$	.994	.174	$\log(P)$	.981	.000
MVP2	$\log(Q)$	.996	.437	$\log(P)$	.960	.000
MVP3	$\log(Q)$	.904	.000	$\log(P)$	.937	.000
MVP4	$\log(Q)$	.944	.000	$\log(P)$	.925	.000

Table B.2. **P-values for Nonparametric Tests – Performance Measure Q.**

Test		RS			SRCH			Noise	
Problem	KW	BM	VDW	WIL	MED	VDW	WIL	MED	VDW
3	.038	$\boxed{.498}$	.026	.000	.000	.000	.000	.003	.000
4	.074	.792	$\boxed{.045}$	.000	.000	.000	.008	$\boxed{.400}$	.002
5	.836	.436	.772	.000	.003	.002	.000	.003	.000
25	.613	.671	.538	.000	.035	.000	.000	.000	.000
36	$\boxed{.051}$	$\boxed{.108}$	$\boxed{.059}$	.000	.000	.000	.003	.006	.002
105	.707	$\boxed{.007}$	.814	.000	.000	.000	.000	.000	.000
110	.322	.532	.284	.977	.293	.768	.000	.000	.000
118	.700	.532	.754	.001	.012	.000	.000	.000	.000
224	.488	.436	.585	.717	.833	.471	.000	.000	.000
244	.235	.792	.135	.041	.006	$\boxed{.149}$	.000	.001	.000
256	.007	$\boxed{.072}$	.012	.000	.000	.000	.000	.002	.000
275	.833	.792	.721	.976	.674	.708	.001	$\boxed{.058}$	.000
281	.022	$\boxed{.150}$	.010	.008	$\boxed{1.00}$	.000	.000	.000	.000
287	$\boxed{.489}$	$\boxed{.274}$	$\boxed{.211}$	.003	.006	.017	.000	.000	.000
288	.000	.000	.000	.002	.000	.022	.000	.000	.000
289	$\boxed{.410}$	$\boxed{.967}$	$\boxed{.247}$	.000	.000	.000	.706	.833	.763
297	.000	.008	.000	.000	.000	.000	.000	.000	.000
300	$\boxed{.133}$	$\boxed{.741}$	$\boxed{.132}$	.000	.000	.000	.000	.000	.000
301	.000	$\boxed{.532}$	.000	$\boxed{.255}$	$\boxed{.528}$	$\boxed{.222}$	.000	.000	.000
305	.000	.027	.000	$\boxed{.456}$	$\boxed{.400}$	$\boxed{.553}$	.000	.000	.000
314	.028	.012	.037	$\boxed{.085}$	$\boxed{.141}$	.050	.000	.002	.000
392	.079	$\boxed{.027}$	.083	.000	.000	.000	.000	.000	.000
MVP1	.000	.000	.000	$\boxed{.023}$	$\boxed{.021}$	.083	.000	.000	.000
MVP2	.000	.000	.000	.759	.599	.540	.000	.002	.000
MVP3	.000	.002	.000	.765	.833	.846	.000	.000	.000
MVP4	.004	$\boxed{.967}$	.000	$\boxed{.276}$	$\boxed{.674}$	$\boxed{.435}$	.000	.000	.000

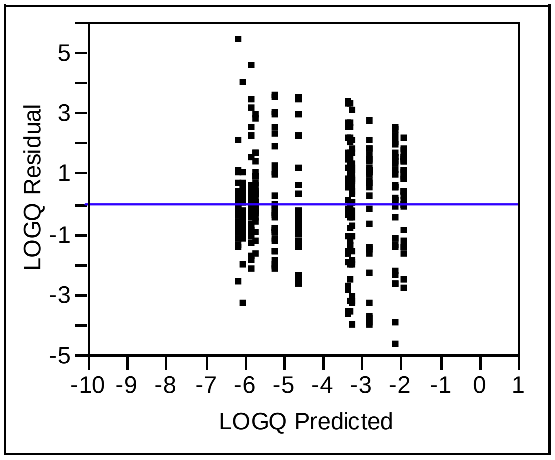
Table B.3. **P-values for Nonparametric Tests – Performance Measure *P*.**

Test		RS			SRCH			Noise	
Problem	KW	BM	VDW	WIL	MED	VDW	WIL	MED	VDW
3	.470	.292	.424	.000	.000	.000	.760	.674	.601
4	.055	.792	$\boxed{.032}$	.000	.000	.000	.039	$\boxed{.400}$	.016
5	.734	.876	.625	.064	.207	.120	.000	.012	.000
25	.653	.967	.485	.000	.000	.000	.000	.000	.037
36	.491	.792	.485	$\boxed{.086}$	$\boxed{1.00}$	.032	.024	$\boxed{.528}$	.017
105	.507	.088	.420	.000	.000	.000	.000	.000	.000
110	.318	.532	.281	.932	.293	.798	.000	.000	.000
118	.838	.532	.922	.000	.000	.000	.000	.000	.000
224	.128	.274	.160	.730	.833	.508	.000	.000	.000
244	.867	.875	.815	.000	.000	.000	$\boxed{.080}$	$\boxed{.207}$	.039
256	.076	.108	.134	$\boxed{.210}$	$\boxed{.400}$	$\boxed{.063}$	.000	.000	.000
275	.817	.741	.886	.499	.528	.426	.841	.528	.786
281	.034	$\boxed{.292}$	.007	.817	.833	.859	.000	.000	.000
287	$\boxed{.062}$	$\boxed{.080}$	.040	.000	.000	.000	.000	$\boxed{.674}$	.000
288	.000	$\boxed{.088}$	.000	$\boxed{.105}$	.027	$\boxed{.436}$	.000	.000	.000
289	$\boxed{.410}$	$\boxed{.967}$	$\boxed{.247}$	.000	.000	.000	.706	.833	.763
297	.000	.000	.000	.346	.599	.391	.000	.000	.000
300	.586	.741	.723	.000	.000	.000	.000	.006	.000
301	.004	$\boxed{.357}$	.000	.491	.528	.587	.000	.000	.000
305	.000	.024	.000	$\boxed{.314}$	$\boxed{.141}$	$\boxed{.477}$	.000	.000	.000
314	.087	.357	.112	$\boxed{.030}$	.141	$\boxed{.042}$	.000	.000	.000
392	.052	$\boxed{.039}$	$\boxed{.045}$	.000	.000	.000	.000	.000	.000
MVP1	$\boxed{.067}$	$\boxed{.240}$	.041	.317	.141	.436	.000	.001	.000
MVP2	.000	.000	.000	.589	.204	.737	.000	.003	.000
MVP3	.001	.028	.000	.339	.833	.107	.000	.000	.000
MVP4	$\boxed{.078}$	$\boxed{.967}$	.019	.682	.400	.328	.000	.000	.000

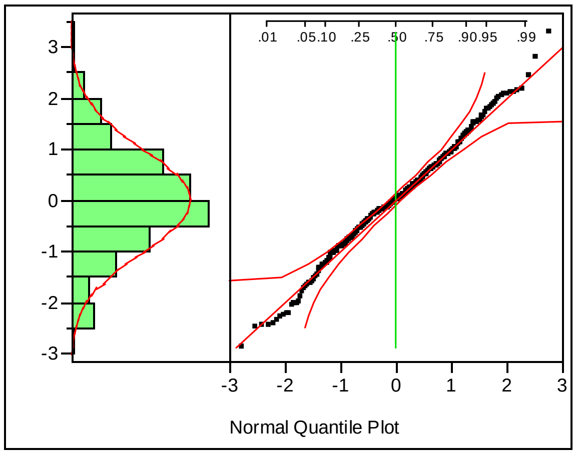
Test Problem 3

Performance Measure Q

Residual vs. Predicted Plot

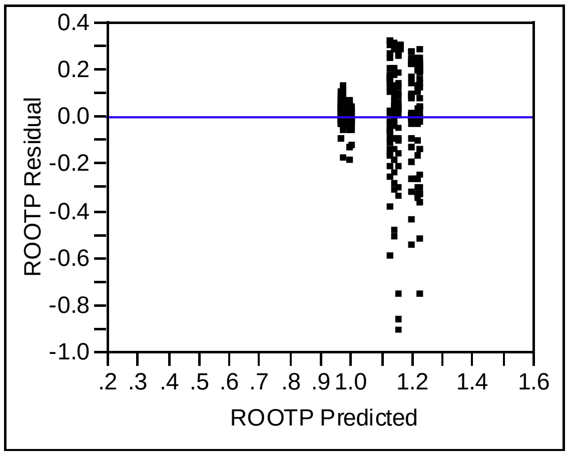


Normal Probability Plot

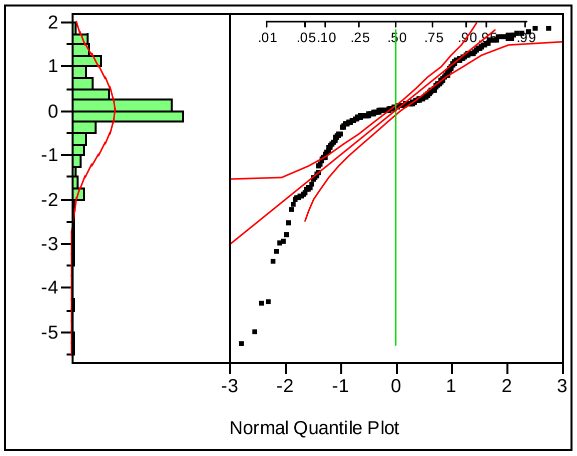


Performance Measure P

Residual vs. Predicted Plot



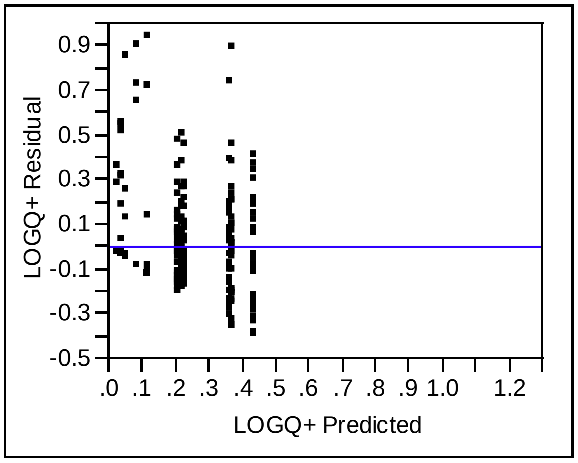
Normal Probability Plot



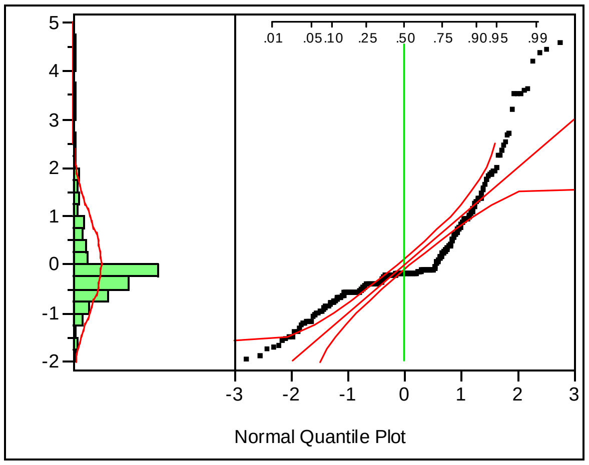
Test Problem 4

Performance Measure Q

Residual vs. Predicted Plot

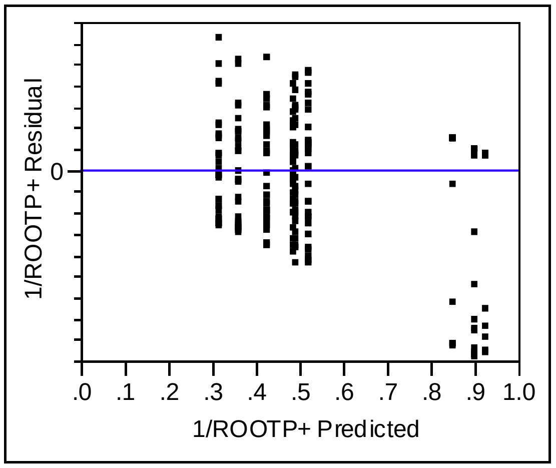


Normal Probability Plot

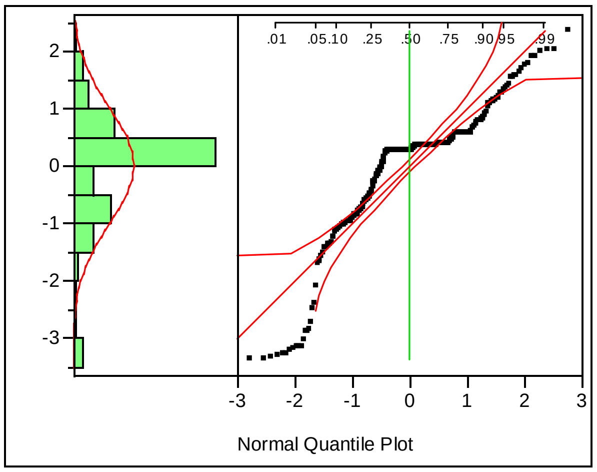


Performance Measure P

Residual vs. Predicted Plot



Normal Probability Plot

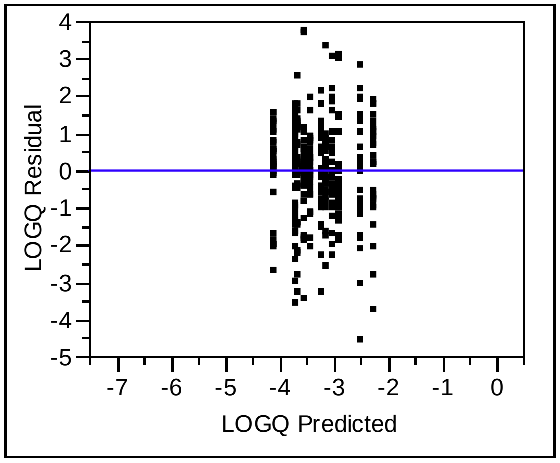


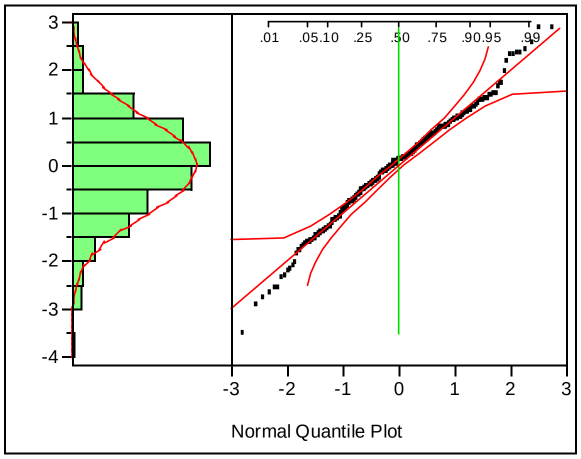
Test Problem 5

Performance Measure Q

Residual vs. Predicted Plot

Normal Probability Plot

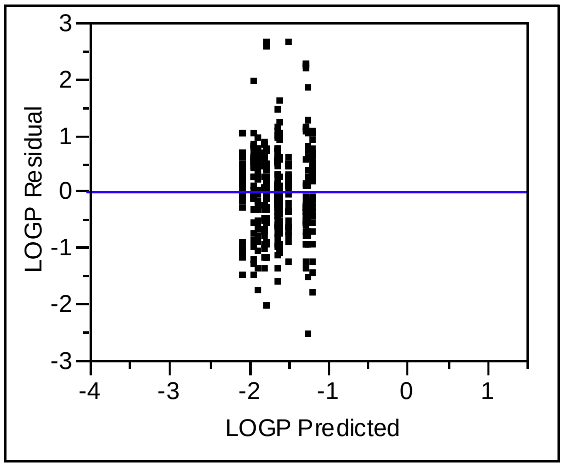


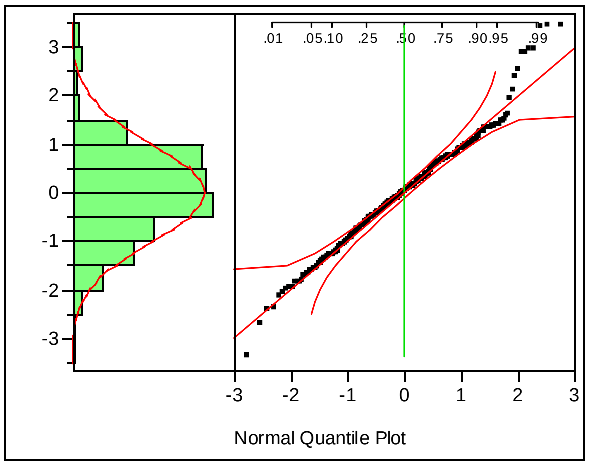


Performance Measure P

Residual vs. Predicted Plot

Normal Probability Plot

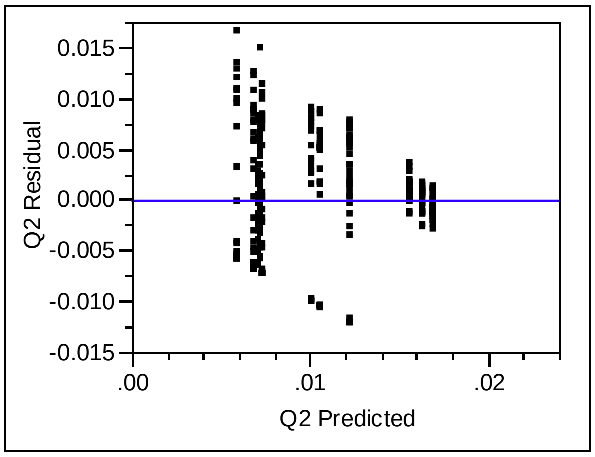




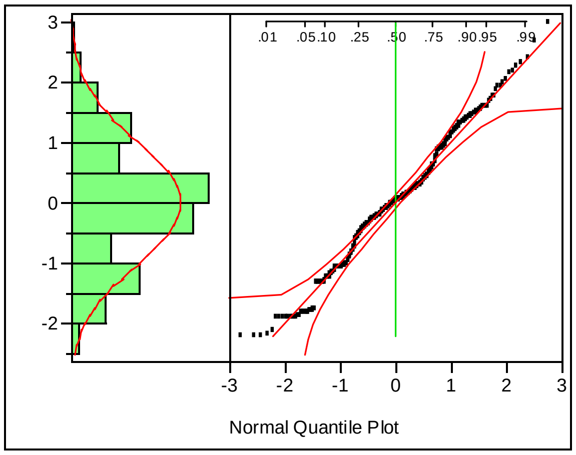
Test Problem 25

Performance Measure Q

Residual vs. Predicted Plot

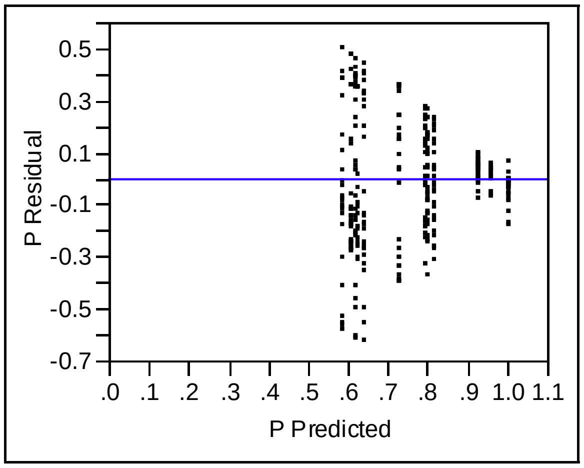


Normal Probability Plot

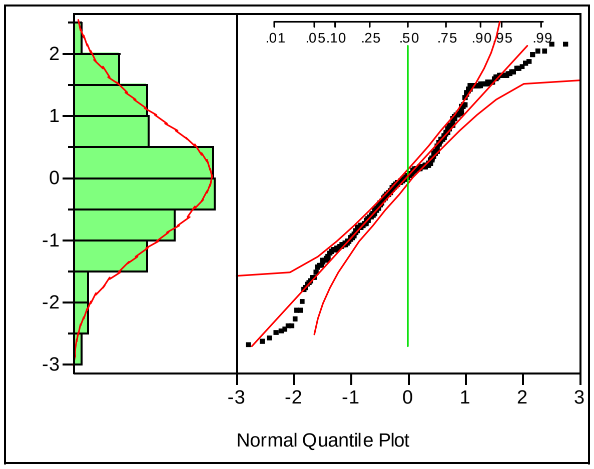


Performance Measure P

Residual vs. Predicted Plot



Normal Probability Plot

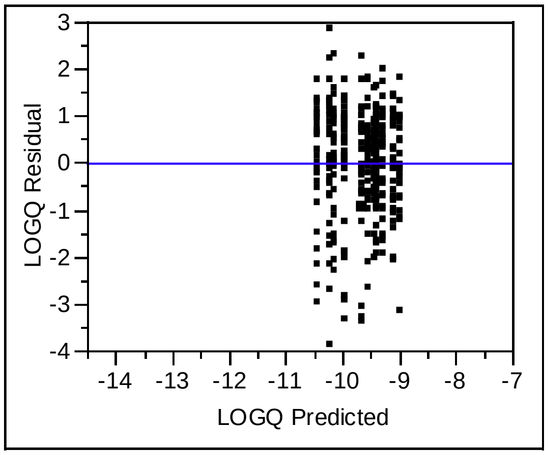


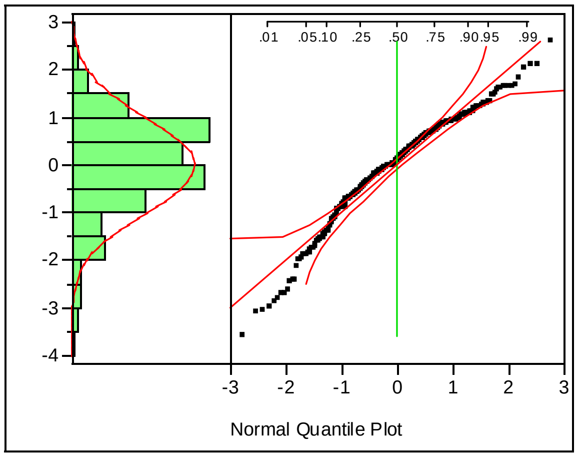
Test Problem 36

Performance Measure Q

Residual vs. Predicted Plot

Normal Probability Plot

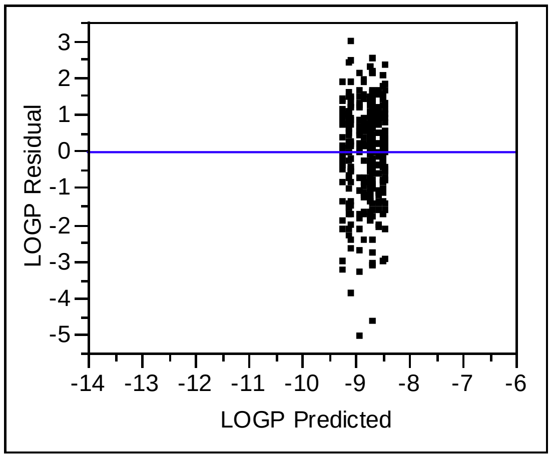


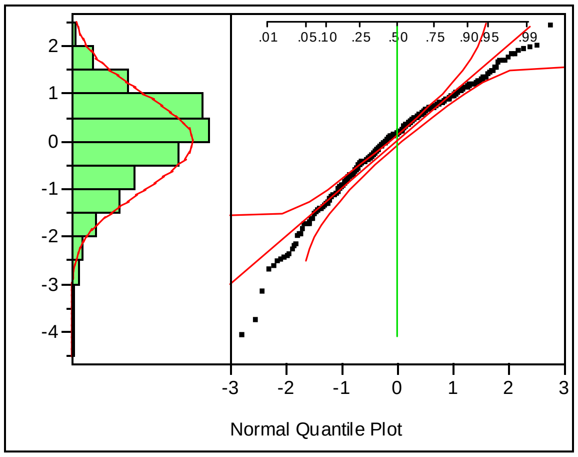


Performance Measure P

Residual vs. Predicted Plot

Normal Probability Plot



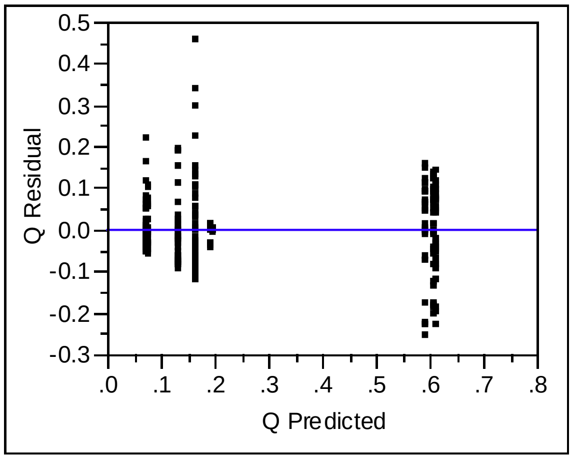


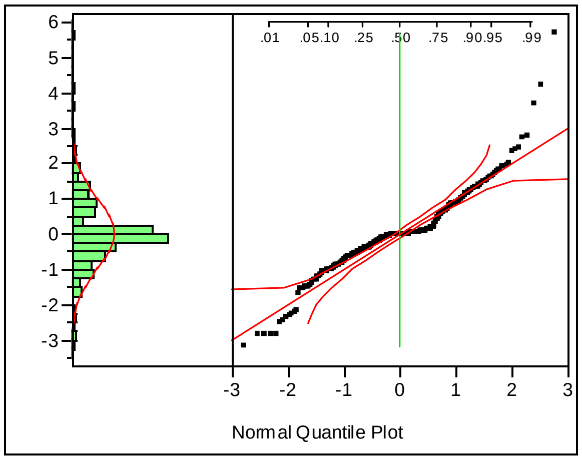
Test Problem 105

Performance Measure Q

Residual vs. Predicted Plot

Normal Probability Plot

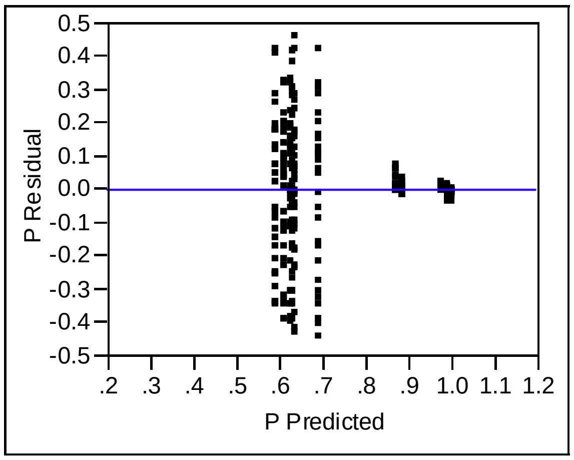


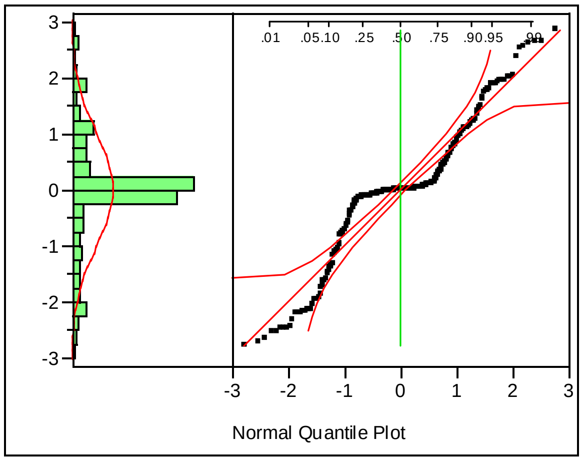


Performance Measure P

Residual vs. Predicted Plot

Normal Probability Plot

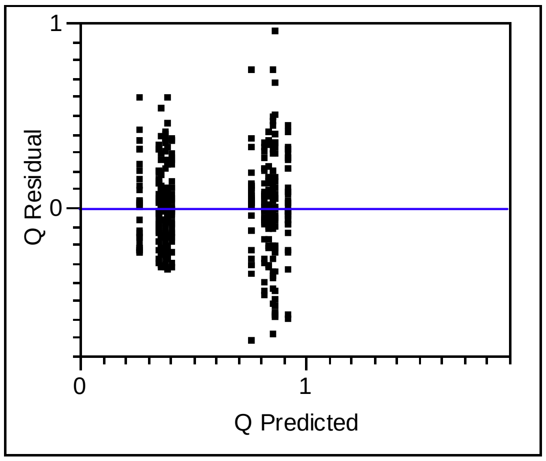




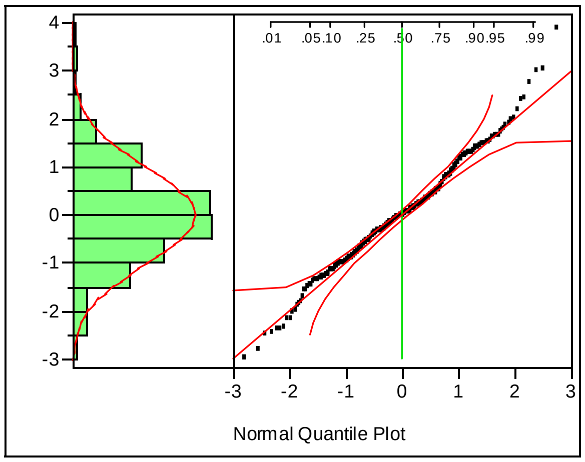
Test Problem 110

Performance Measure Q

Residual vs. Predicted Plot

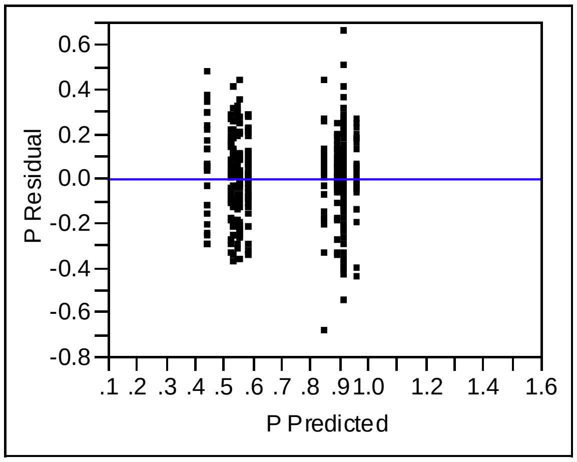


Normal Probability Plot

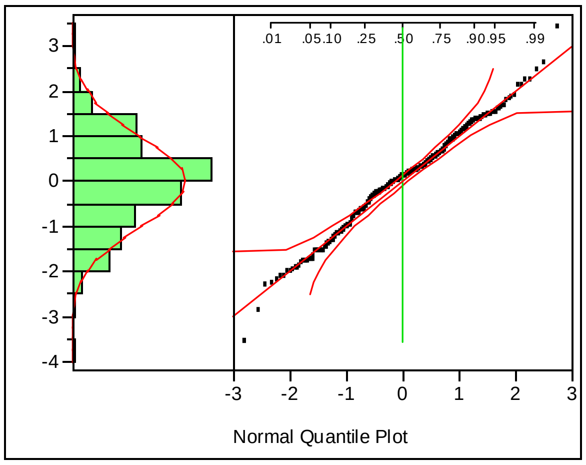


Performance Measure P

Residual vs. Predicted Plot



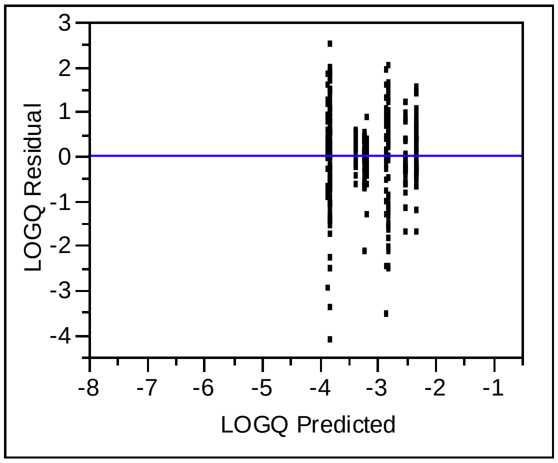
Normal Probability Plot



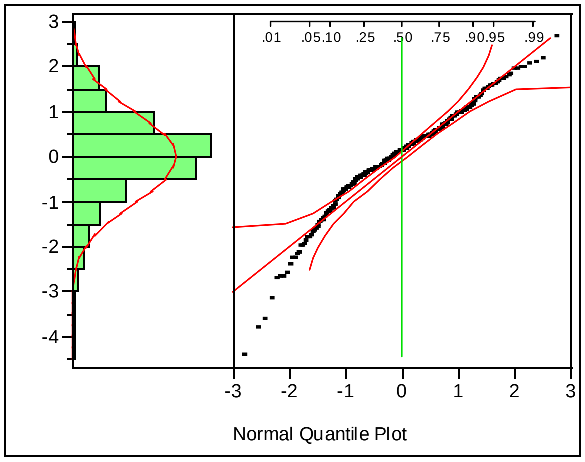
Test Problem 118

Performance Measure Q

Residual vs. Predicted Plot

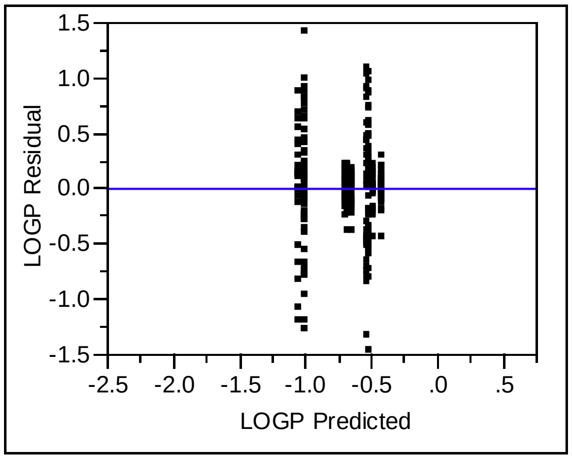


Normal Probability Plot

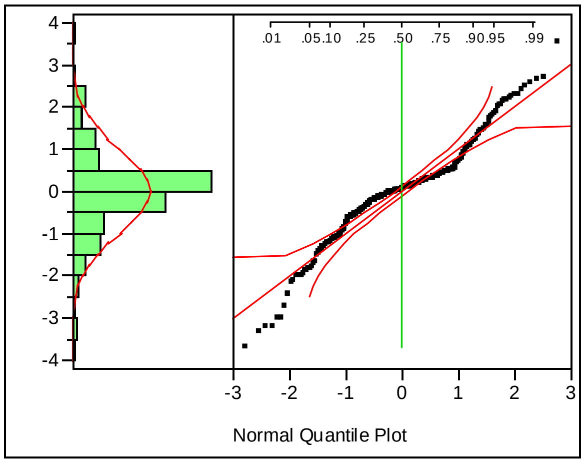


Performance Measure P

Residual vs. Predicted Plot



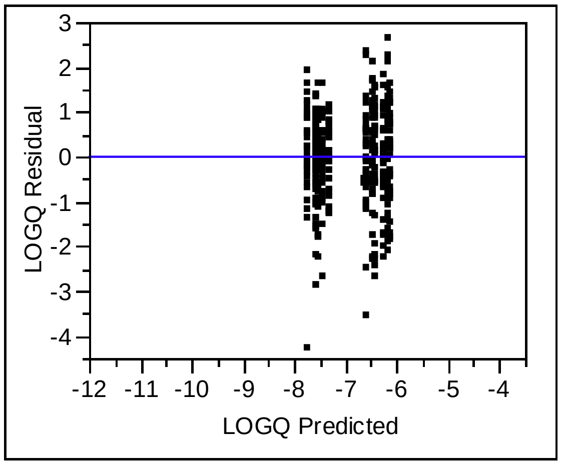
Normal Probability Plot

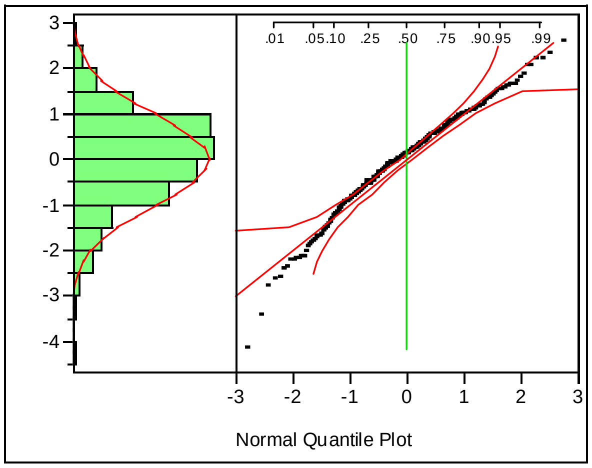


Test Problem 224

Performance Measure Q

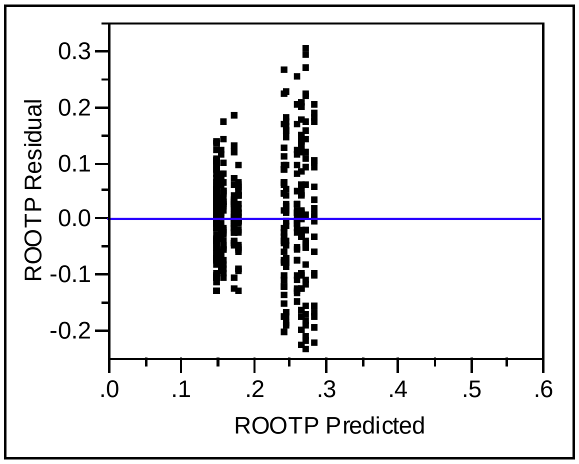
Residual vs. Predicted Plot	Normal Probability Plot

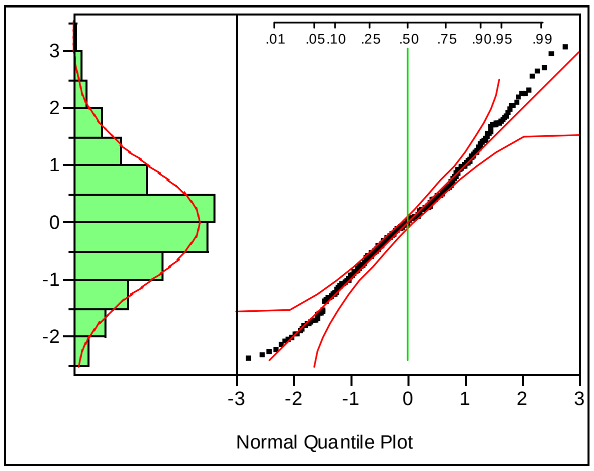




Performance Measure P

Residual vs. Predicted Plot	Normal Probability Plot

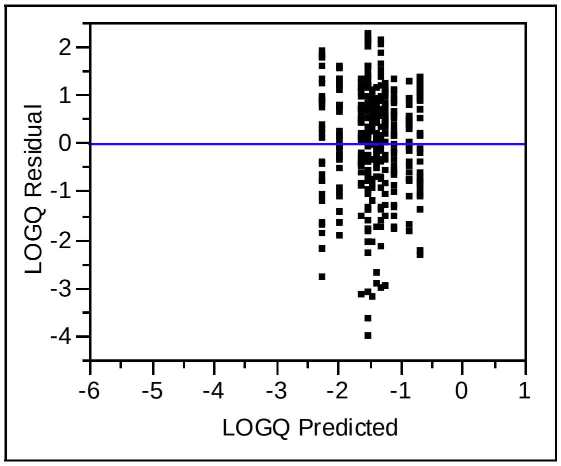




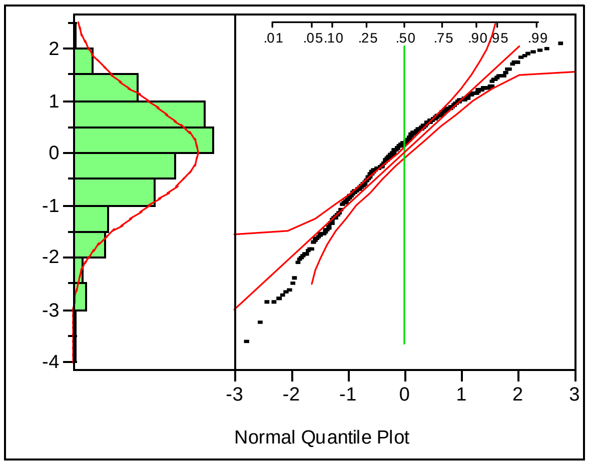
Test Problem 244

Performance Measure Q

Residual vs. Predicted Plot

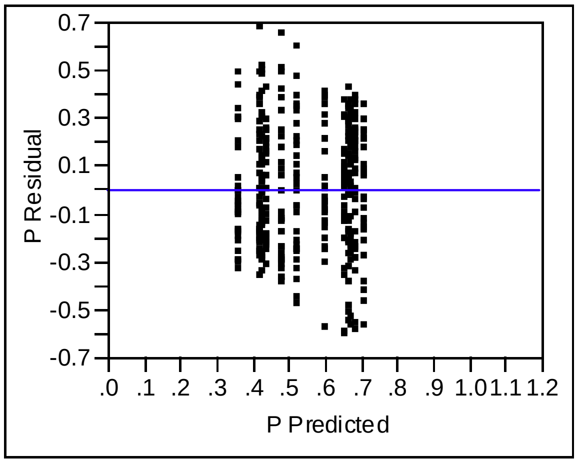


Normal Probability Plot

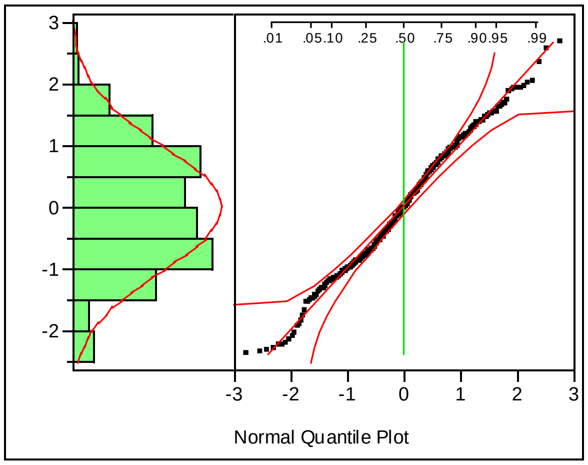


Performance Measure P

Residual vs. Predicted Plot



Normal Probability Plot

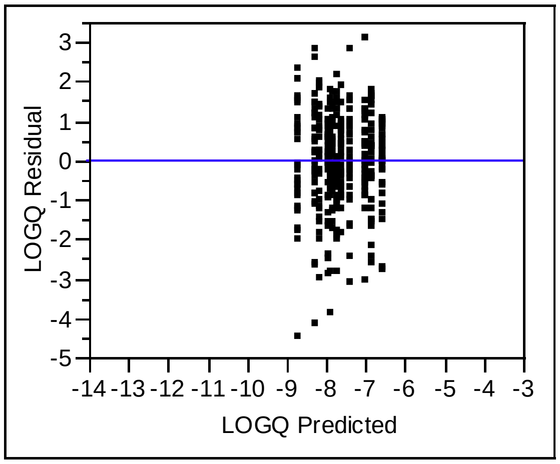


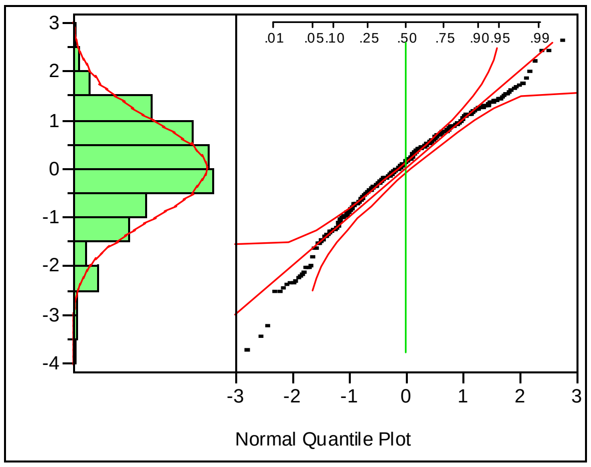
Test Problem 256

Performance Measure Q

Residual vs. Predicted Plot

Normal Probability Plot

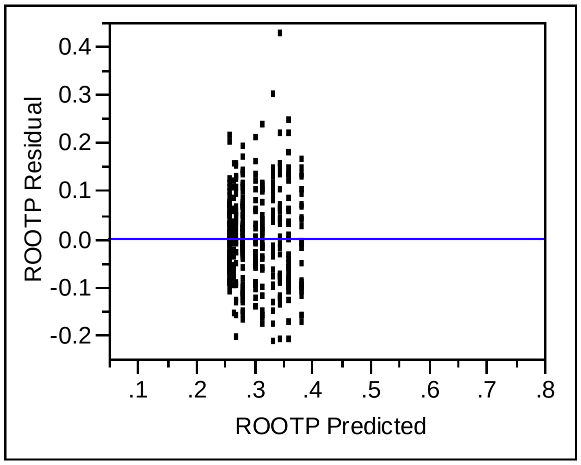


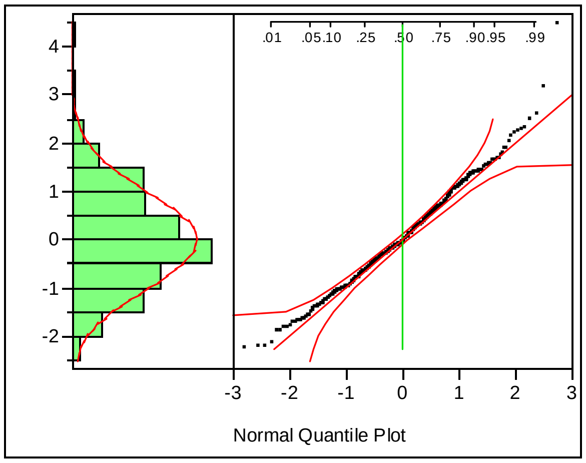


Performance Measure P

Residual vs. Predicted Plot

Normal Probability Plot

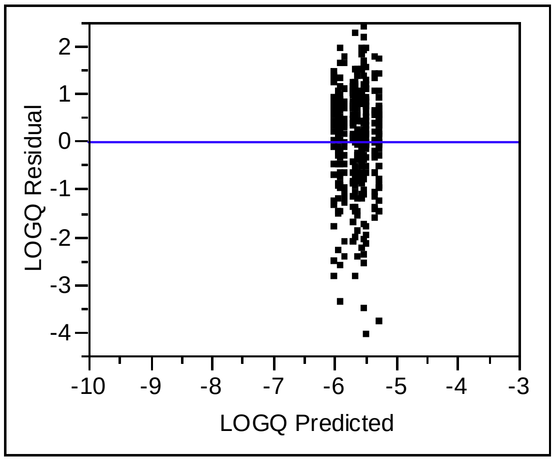




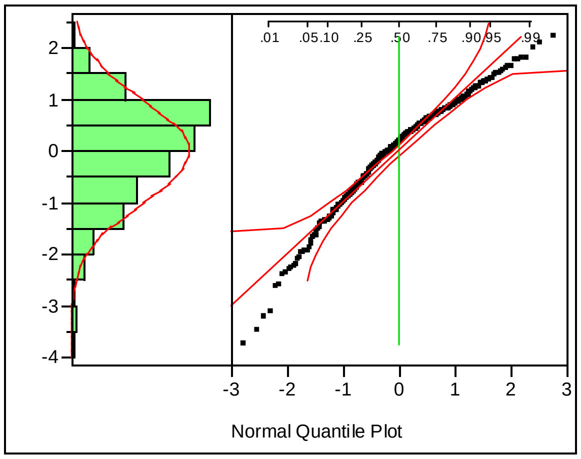
Test Problem 275

Performance Measure Q

Residual vs. Predicted Plot

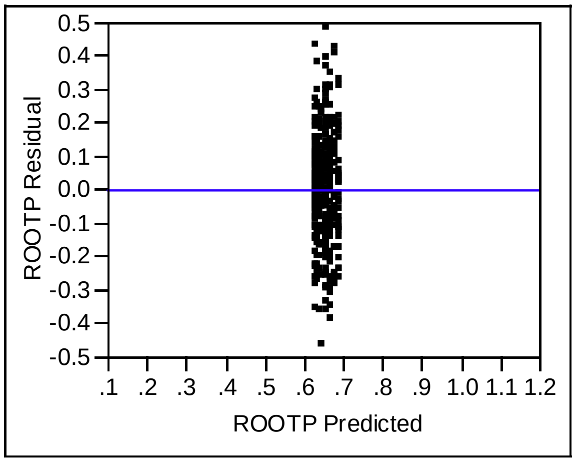


Normal Probability Plot

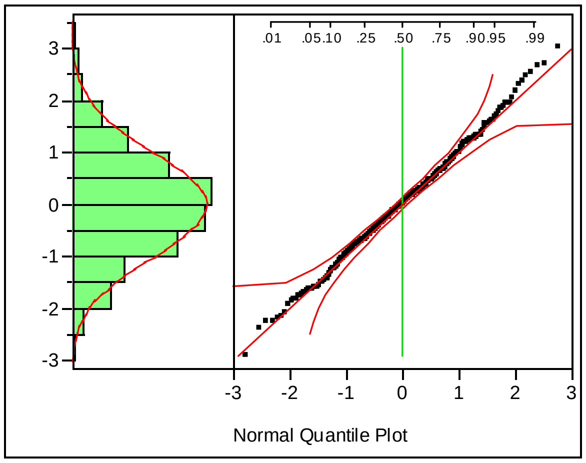


Performance Measure P

Residual vs. Predicted Plot



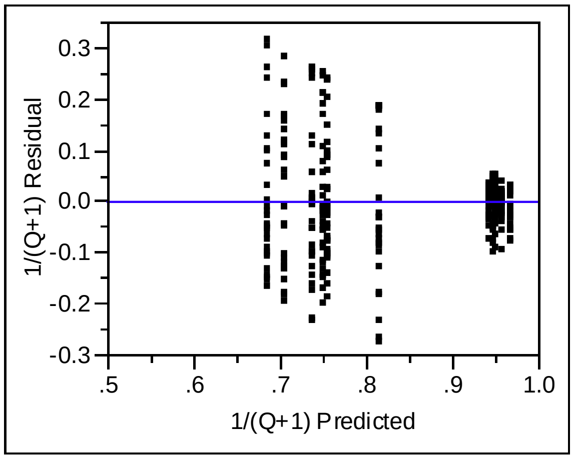
Normal Probability Plot



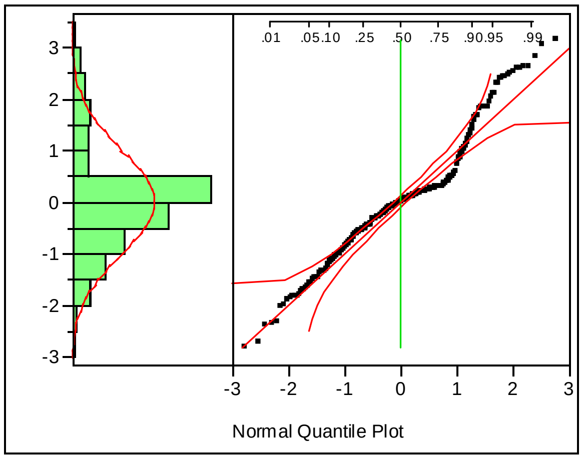
Test Problem 281

Performance Measure Q

Residual vs. Predicted Plot

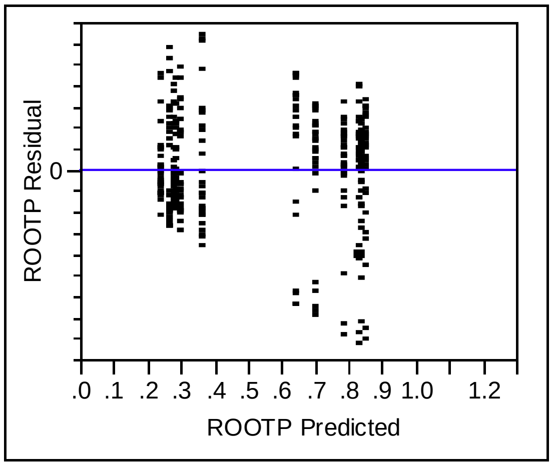


Normal Probability Plot

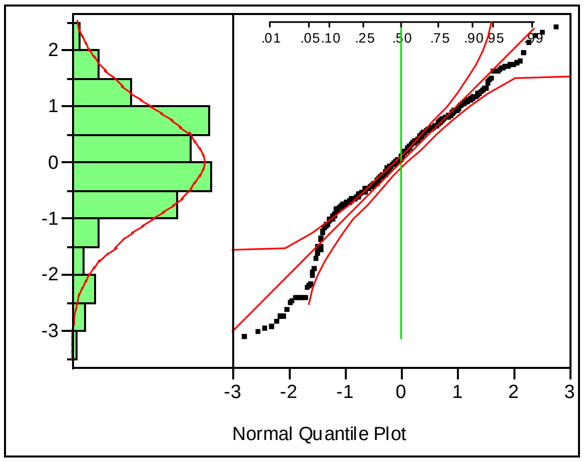


Performance Measure P

Residual vs. Predicted Plot



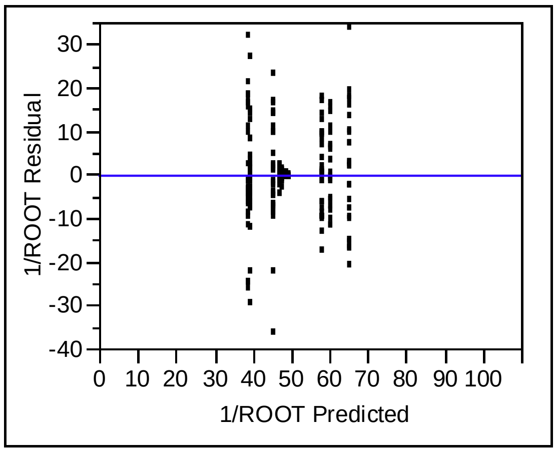
Normal Probability Plot



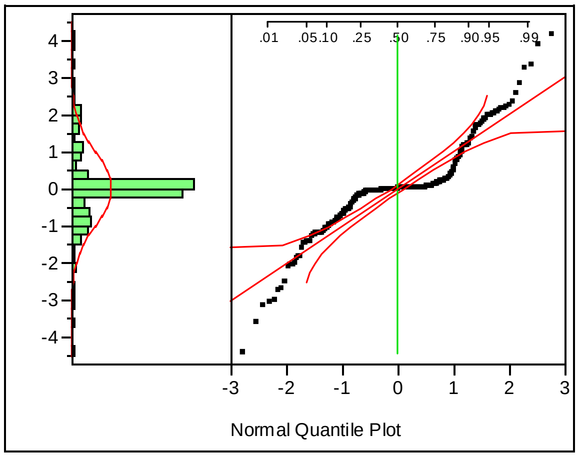
Test Problem 287

Performance Measure Q

Residual vs. Predicted Plot

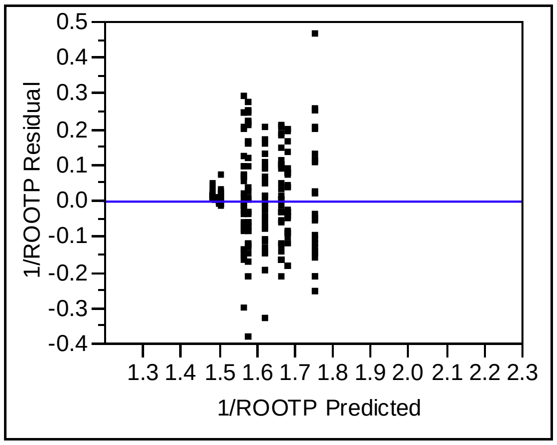


Normal Probability Plot

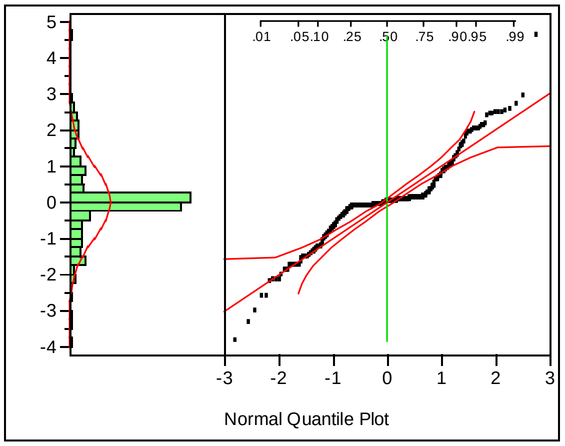


Performance Measure P

Residual vs. Predicted Plot



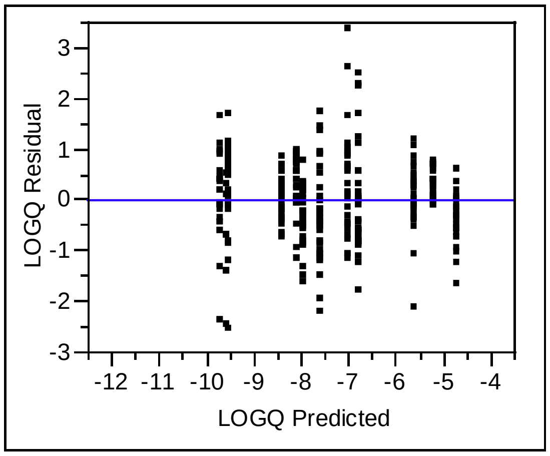
Normal Probability Plot



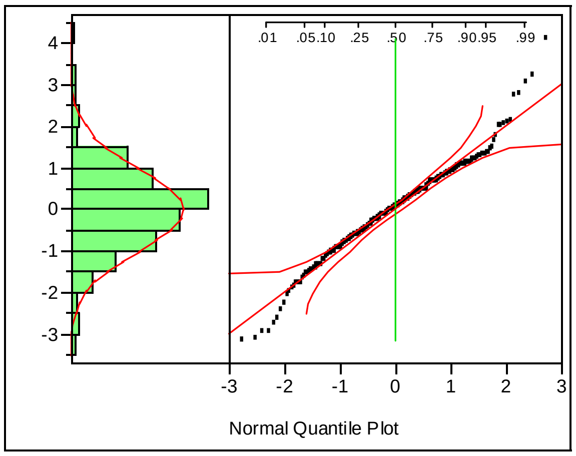
Test Problem 288

Performance Measure Q

Residual vs. Predicted Plot

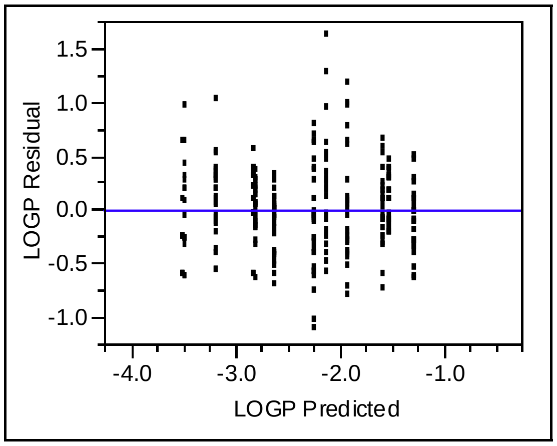


Normal Probability Plot

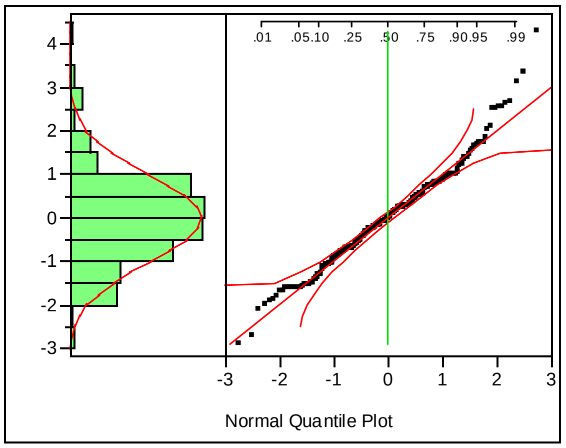


Performance Measure P

Residual vs. Predicted Plot



Normal Probability Plot

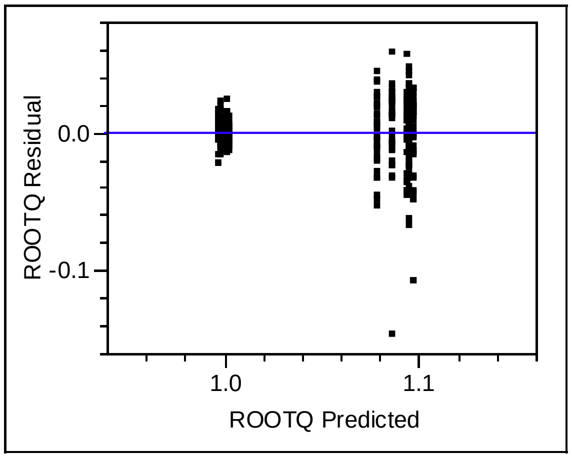


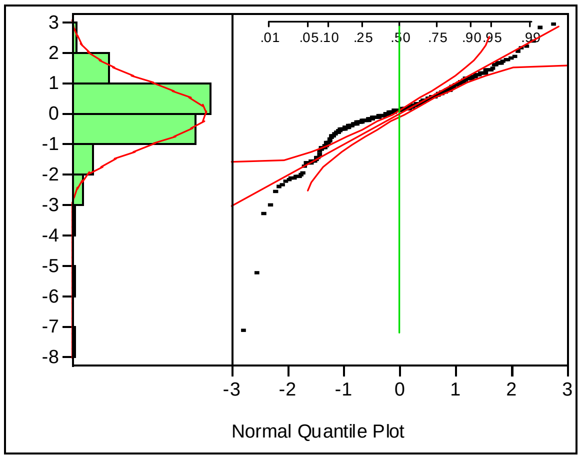
Test Problem 289

Performance Measure Q

Residual vs. Predicted Plot

Normal Probability Plot

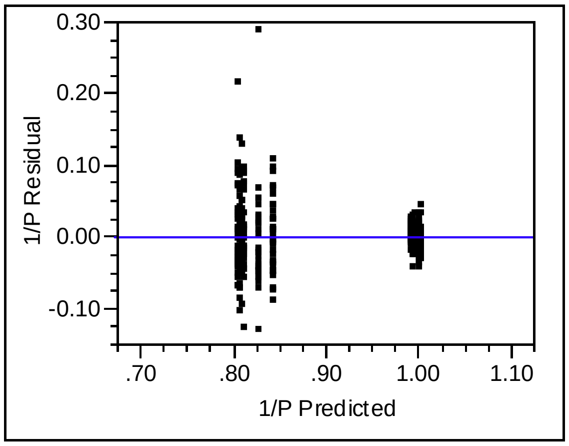


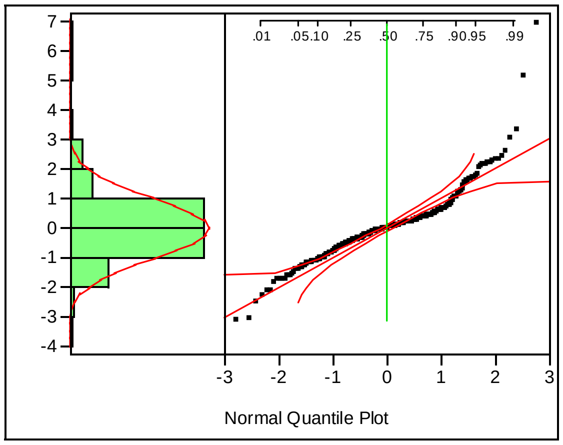


Performance Measure P

Residual vs. Predicted Plot

Normal Probability Plot

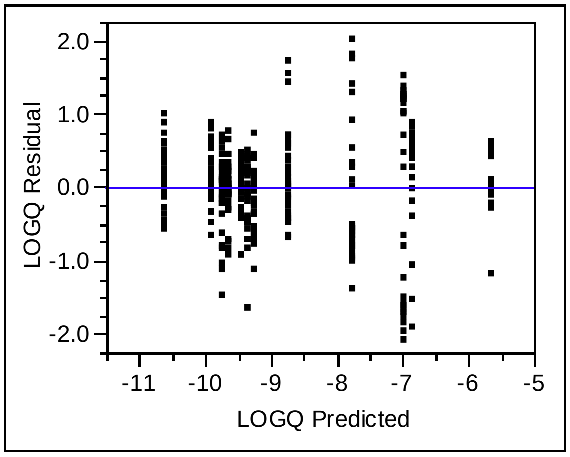




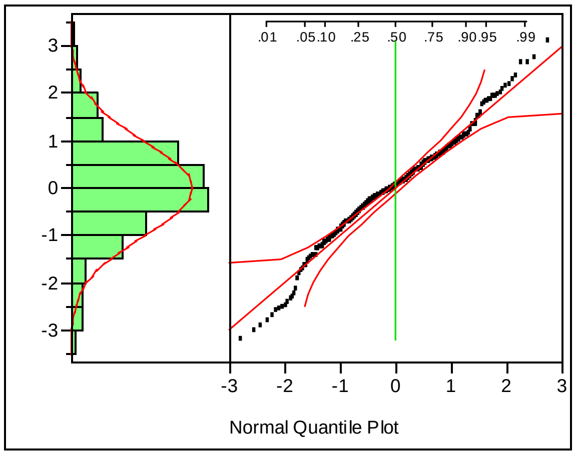
Test Problem 297

Performance Measure Q

Residual vs. Predicted Plot

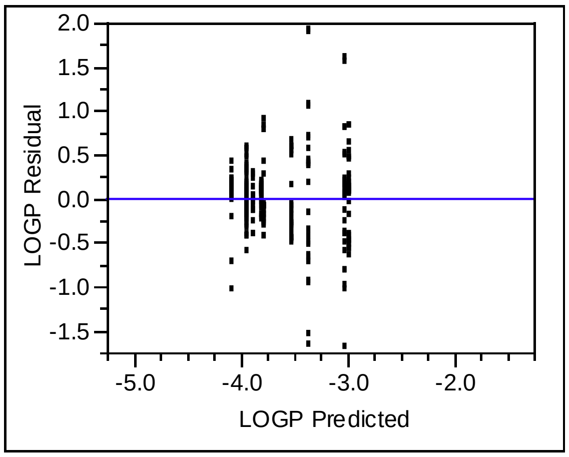


Normal Probability Plot

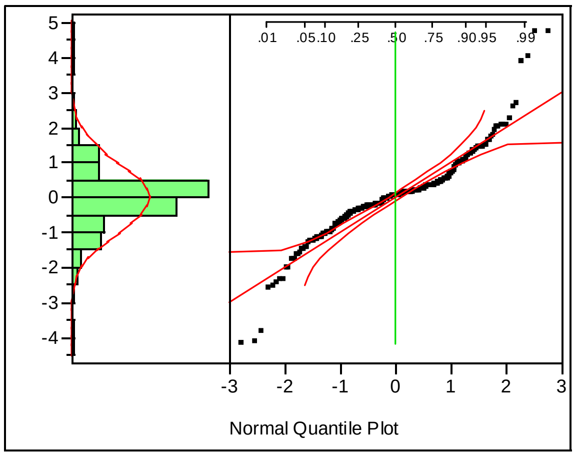


Performance Measure P

Residual vs. Predicted Plot



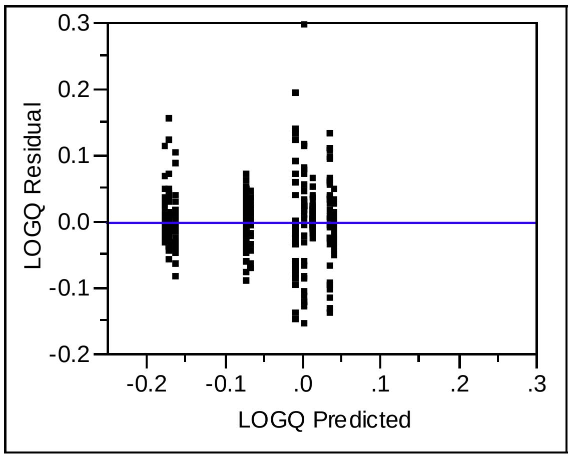
Normal Probability Plot



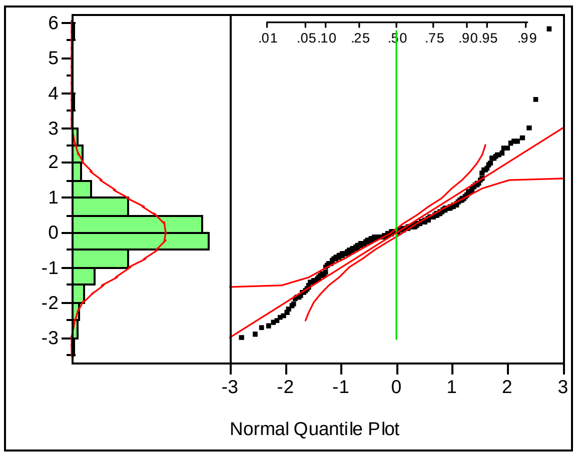
Test Problem 300

Performance Measure Q

Residual vs. Predicted Plot

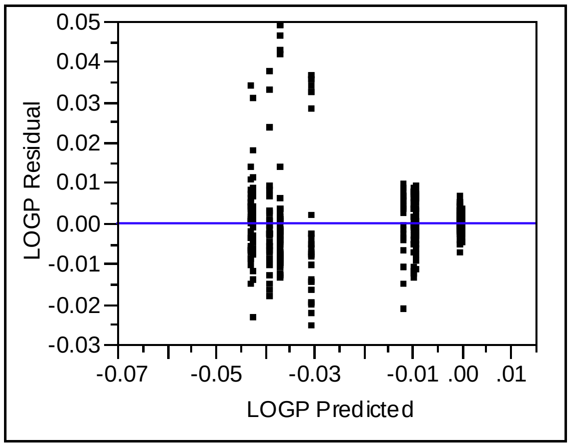


Normal Probability Plot

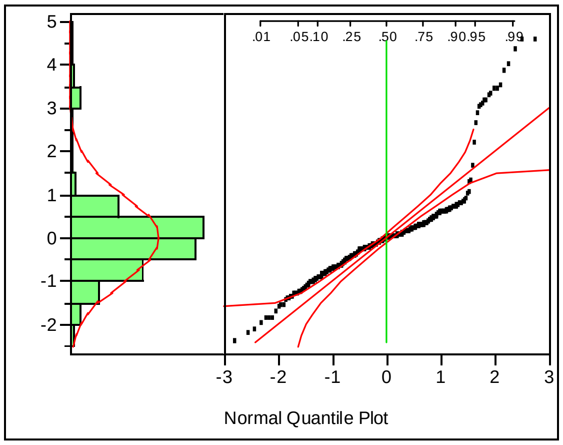


Performance Measure P

Residual vs. Predicted Plot



Normal Probability Plot

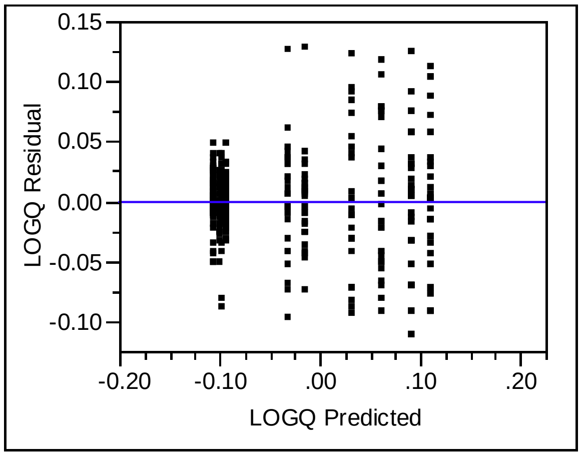


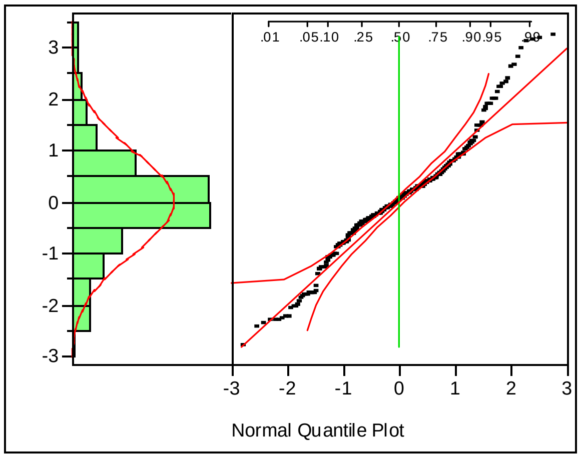
Test Problem 301

Performance Measure Q

Residual vs. Predicted Plot

Normal Probability Plot

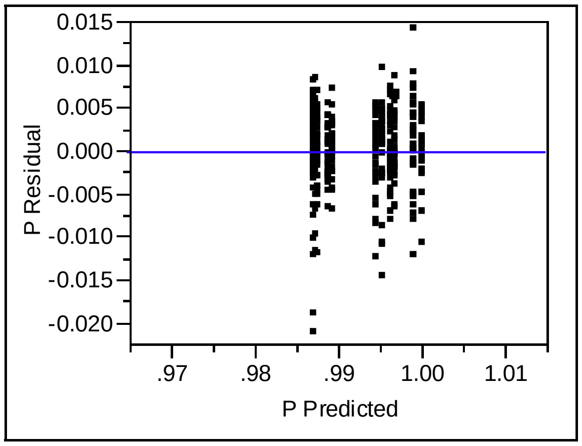


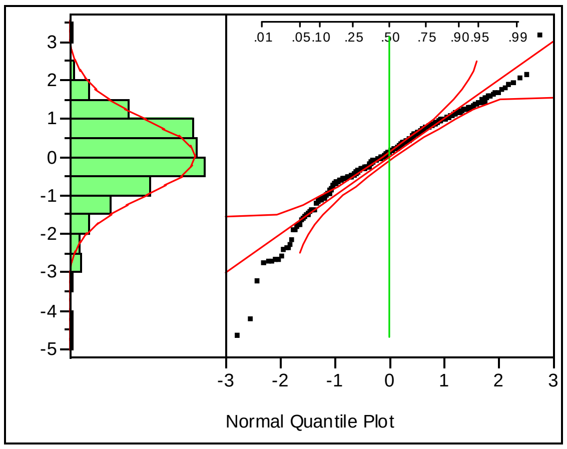


Performance Measure P

Residual vs. Predicted Plot

Normal Probability Plot

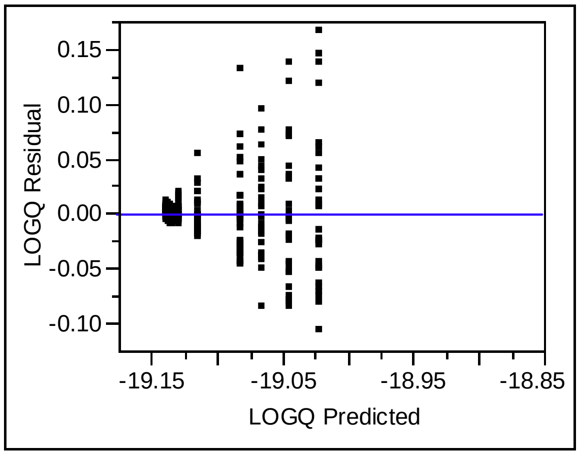




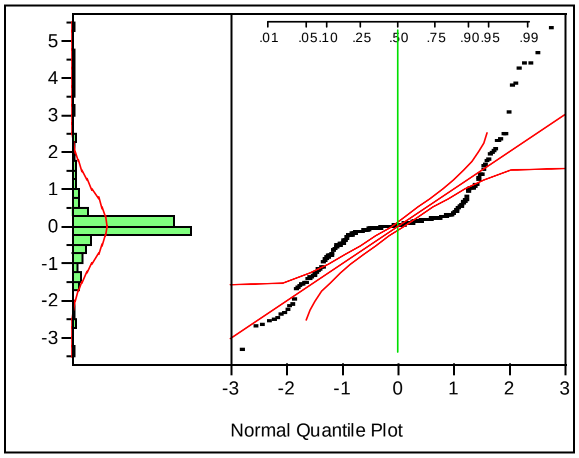
Test Problem 305

Performance Measure Q

Residual vs. Predicted Plot

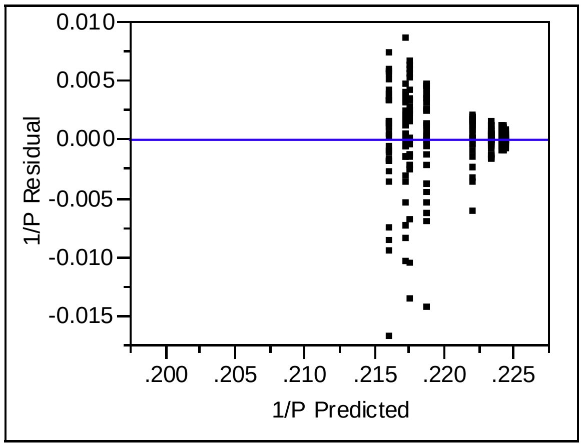


Normal Probability Plot

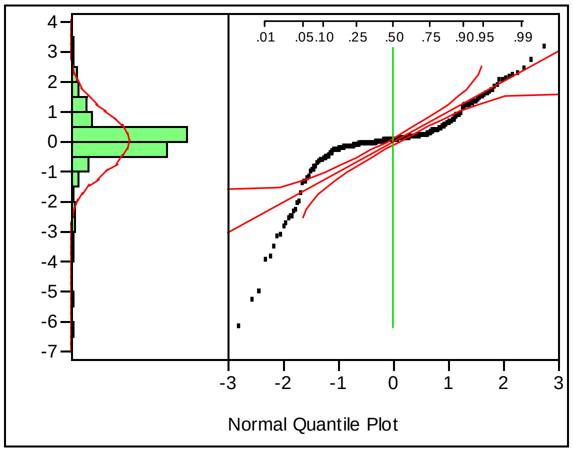


Performance Measure P

Residual vs. Predicted Plot



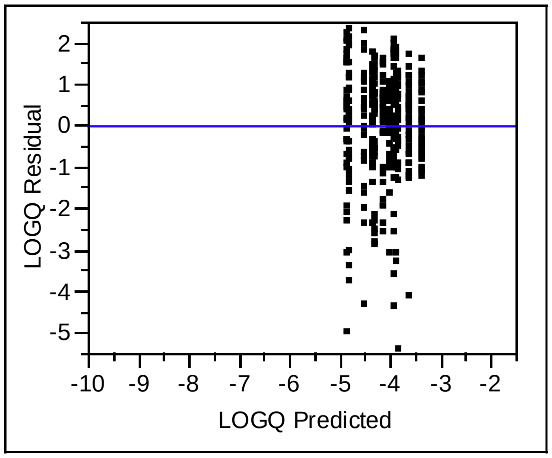
Normal Probability Plot



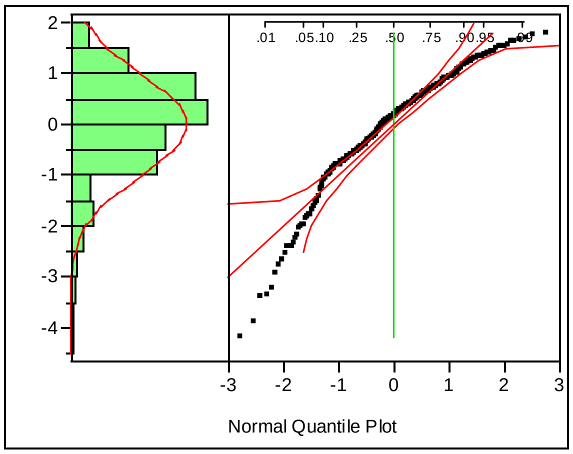
Test Problem 314

Performance Measure Q

Residual vs. Predicted Plot

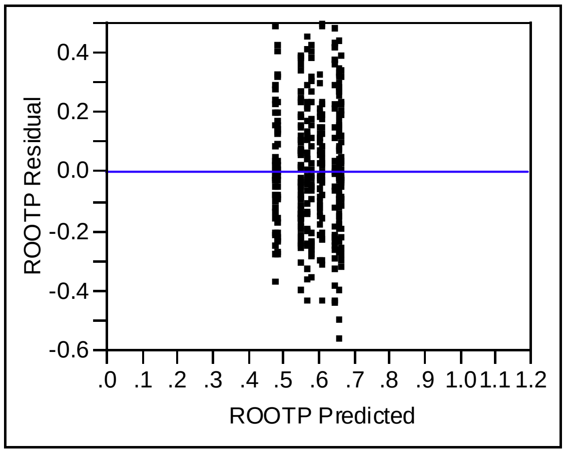


Normal Probability Plot

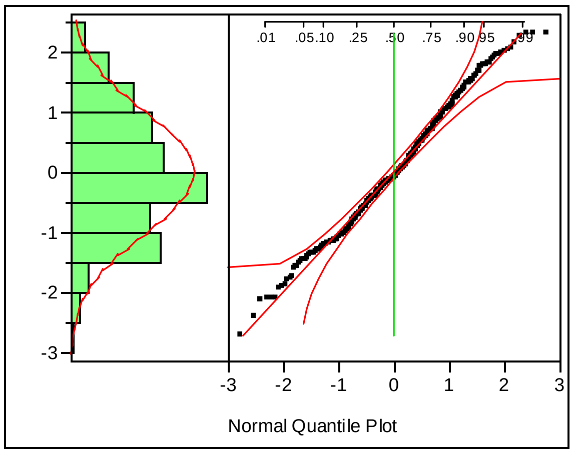


Performance Measure P

Residual vs. Predicted Plot



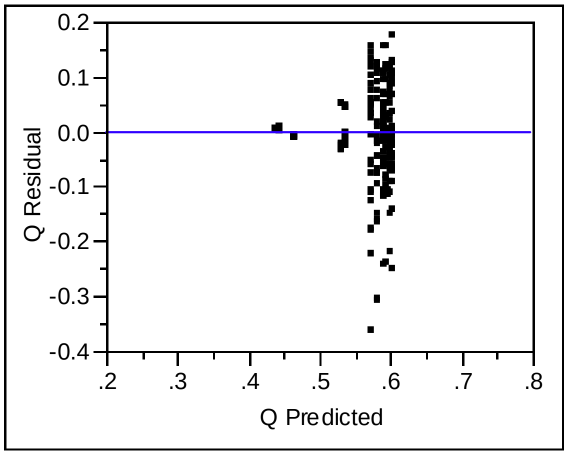
Normal Probability Plot



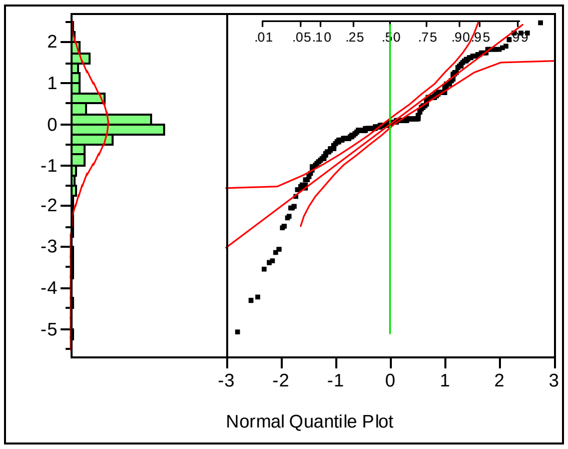
Test Problem 392

Performance Measure Q

Residual vs. Predicted Plot

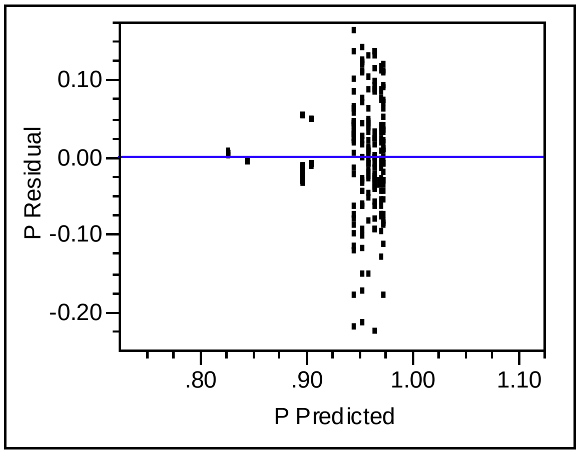


Normal Probability Plot

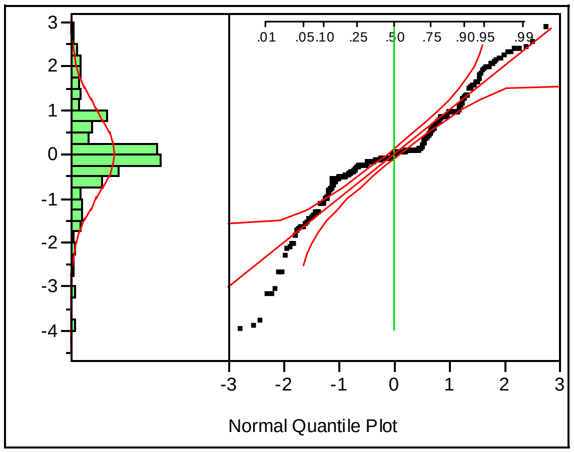


Performance Measure P

Residual vs. Predicted Plot



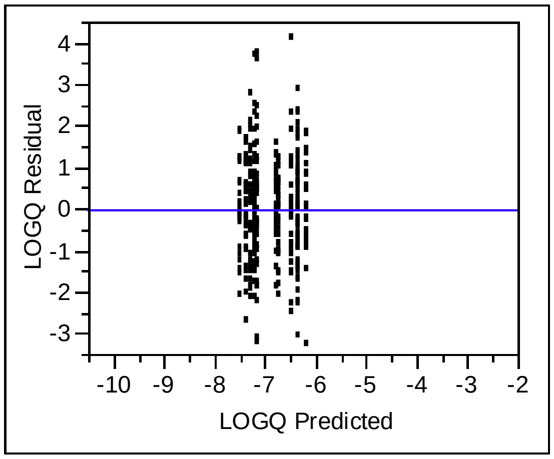
Normal Probability Plot



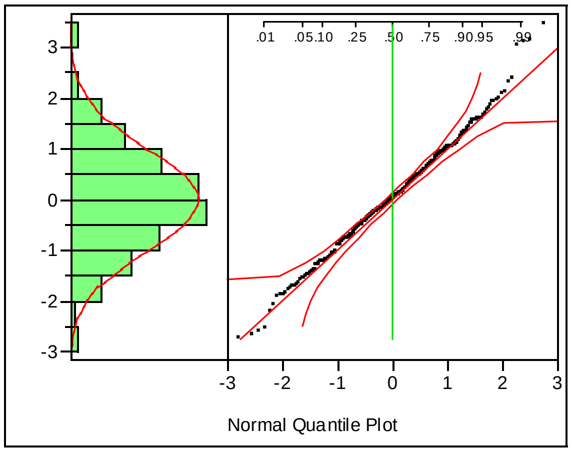
Test Problem MVP1

Performance Measure Q

Residual vs. Predicted Plot

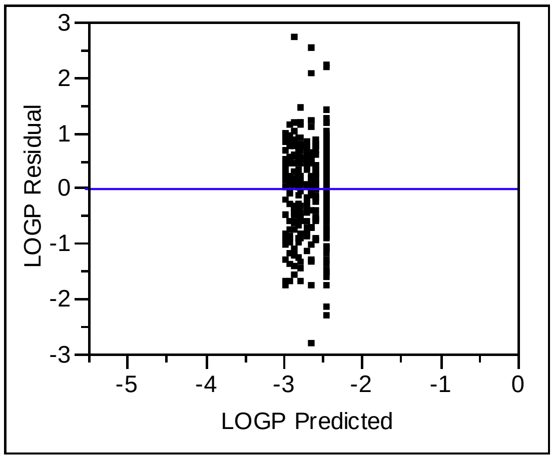


Normal Probability Plot

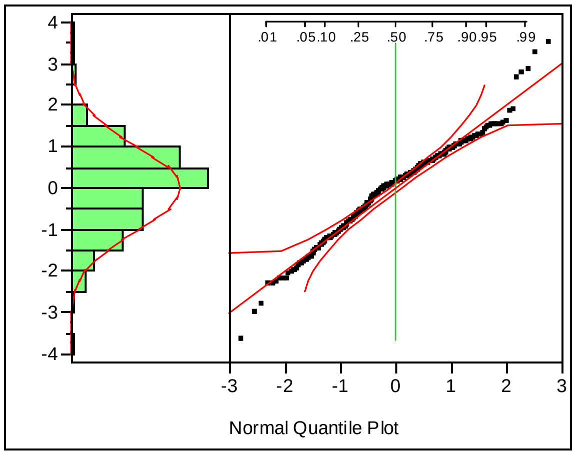


Performance Measure P

Residual vs. Predicted Plot



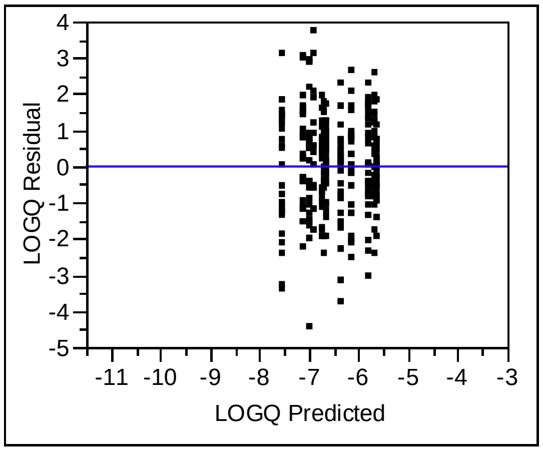
Normal Probability Plot



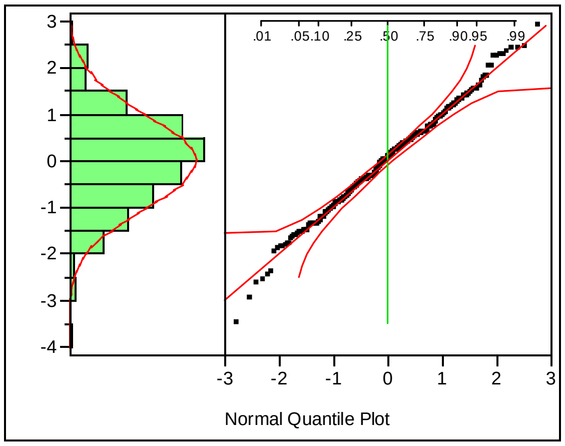
Test Problem MVP2

Performance Measure Q

Residual vs. Predicted Plot

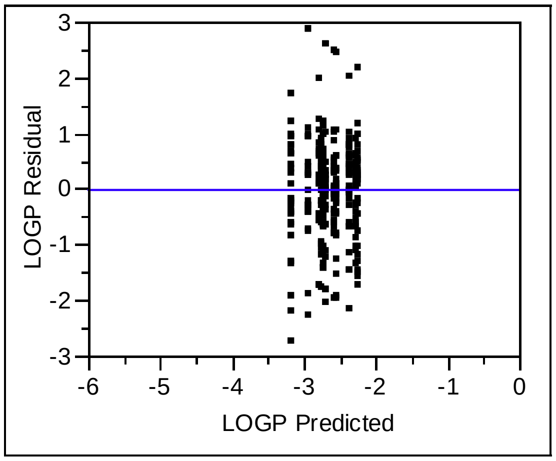


Normal Probability Plot

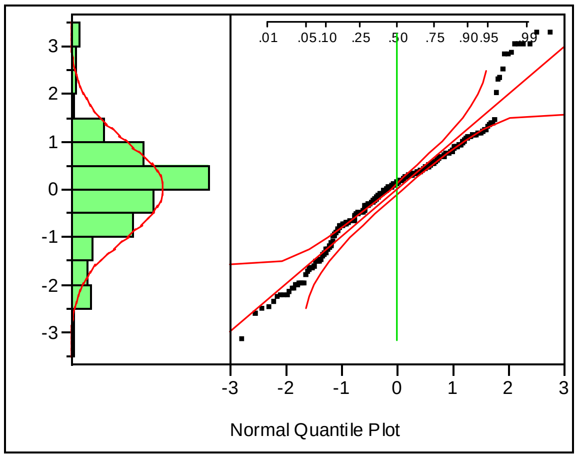


Performance Measure P

Residual vs. Predicted Plot



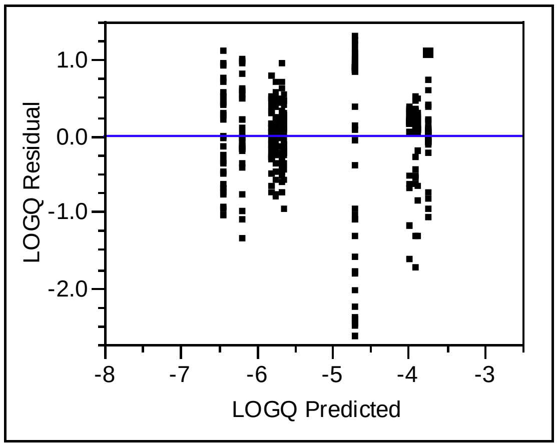
Normal Probability Plot



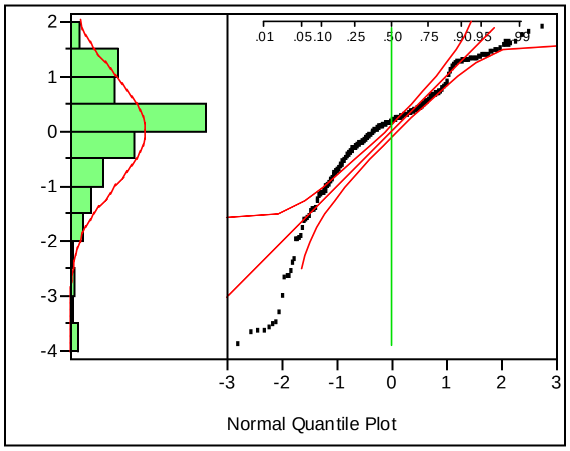
Test Problem MVP3

Performance Measure Q

Residual vs. Predicted Plot

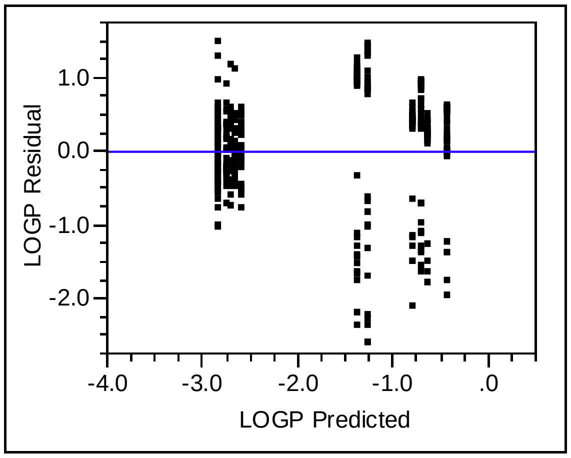


Normal Probability Plot

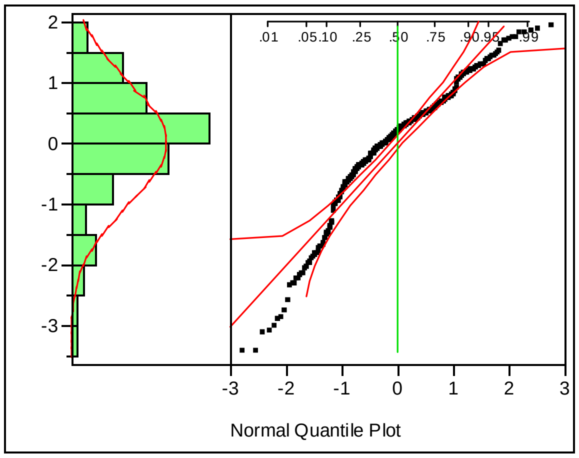


Performance Measure P

Residual vs. Predicted Plot



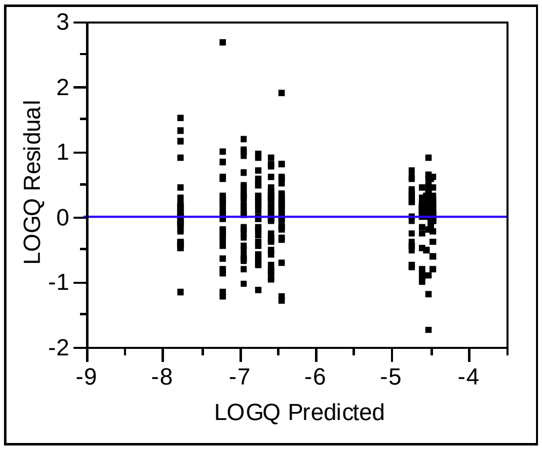
Normal Probability Plot



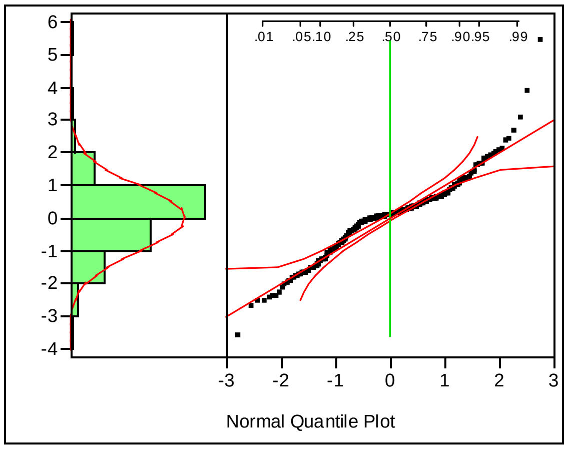
Test Problem MVP4

Performance Measure Q

Residual vs. Predicted Plot

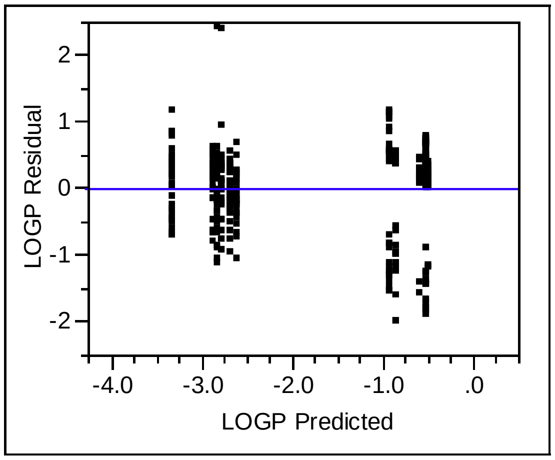


Normal Probability Plot

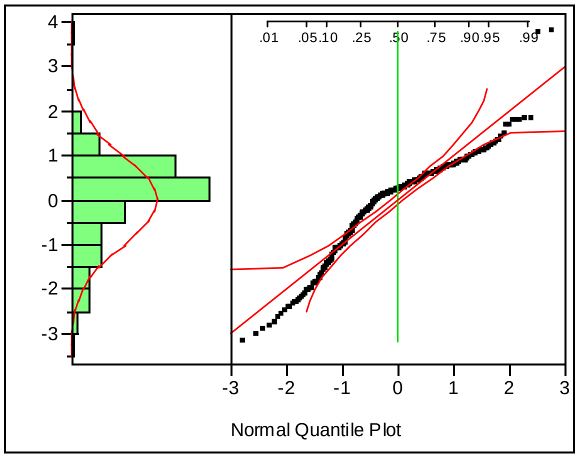


Performance Measure P

Residual vs. Predicted Plot



Normal Probability Plot



BIBLIOGRAPHY

[1] Abramson, M. A. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems.* PhD thesis, Rice University, Department of Computational and Applied Mathematics, 2002. also appears as Tech. Rep. TR02-11.

[2] Abramson, M. A., Audet, C., and Dennis, J. Filter pattern search algorithms for mixed variable constrained optimization problems. Tech. Rep. TR04-09, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, June 2004.

[3] Abspoel, S. J., Etman, L. F. P., Vervoort, J., VanRooij, R. A., Schoofs, A. J. G., and Rooda, J. E. Simulation based optimization of stochastic systems with integer design variables by sequential multipoint linear approximation. *Structural and Multidisciplinary Optimization 22* (2001), 125–138.

[4] Ahmed, M. A., and Alkhamis, T. M. Simulation-based optimization using simulated annealing with ranking and selection. *Computers and Operations Research 29* (2002), 387–402.

[5] Anderberg, M. R. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[6] Anderson, E. J., and Ferris, M. C. A direct search algorithm for optimization with noisy function evaluations. *SIAM Journal on Optimization 11*, 3 (2001), 837–857.

[7] Andradóttir, S. A method for discrete stochastic optimization. *Management Science 41*, 12 (1995), 1946–1961.

[8] Andradóttir, S. A global search method for discrete stochastic optimization. *SIAM Journal on Optimization 6*, 2 (1996), 513–530.

[9] Andradóttir, S. A review of simulation optimization techniques. In *Proceedings of the 1998 Winter Simulation Conference* (Picataway, New Jersey, 1998), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., Institute of Electrical and Electronics Engineers, pp. 151–158.

[10] Andradóttir, S. Simulation optimization (chap. 9). In *Handbook of Simulation* (New York, 1998), J. Banks, Ed., John Wiley and Sons, pp. 307–333.

[11] Andradóttir, S. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM Transactions on Modeling and Computer Simulation 9*, 4 (1999), 349–380.

[12] Angün, E., Kleijnen, J., Hertog, D. D., and G.Gürkan. Response surface methodology revisited. In *Proceedings of the 2002 Winter Simulation Conference* (Piscataway, New Jersey, 2002), E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, Eds., Institute of Electrical and Electronics Engineers, pp. 377–383.

[13] Audet, C., and Dennis, Jr., J. E. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization 11*, 3 (2000), 573–594.

[14] Audet, C., and Dennis, Jr., J. E. Analysis of generalized pattern searches. *SIAM Journal on Optimization 13*, 3 (2003), 889–903.

[15] Audet, C., and Dennis, Jr., J. E. Mesh adaptive direct search algorithms for constrained optimization. Tech. Rep. TR04-02, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, January 2004.

[16] Audet, C., and Dennis, Jr., J. E. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization 14*, 4 (2004), 980–1010.

[17] Azadivar, F. Simulation optimization methodologies. In *Proceedings of the 1999 Winter Simulation Conference* (Piscataway, New Jersey, 1999), P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, Eds., Institute of Electrical and Electronics Engineers, pp. 93–100.

[18] Azadivar, F., and Talavage, J. Optimization of stochastic simulation models. *Mathematics and Computers in Simulation 22* (1980), 231–241.

[19] Baba, N., and Shoman, T. A modified convergence theorem for a random optimization method. *Information Sciences 13* (1977), 159–166.

[20] Barton, R. R. Minimization algorithms for functions with random noise. *American Journal of Mathematical and Management Sciences 4*, 1/2 (1984), 109–138.

[21] Barton, R. R. Simulation metamodels. In *Proceedings of the 1998 Winter Simulation Conference* (Piscataway, New Jersey, 1998), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., Institute of Electrical and Electronics Engineers, pp. 167–174.

[22] Barton, R. R., and Ivey, Jr. , J. S. Modifications of the nelder-mead simplex method for stochastic simulation response optimization. In *Proceedings of the 1991 Winter Simulation Conference* (Piscataway, New Jersey, 1991), B. L. Nelson, W. D. Kelton, and G. M. Clark, Eds., Institute of Electrical and Electronics Engineers, pp. 945–953.

[23] Barton, R. R., and Ivey, Jr., J. S. Nelder-mead simplex modifications for simulation optimization. *Management Science 42*, 7 (1996), 954–973.

[24] Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. *Linear Programming and Network Flows*, 2nd ed. Wiley & Sons, New York, 1990.

[25] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. *Nonlinear Programming: Theory and Algorithms*, 2nd ed. Wiley & Sons, New York, 1993.

[26] Bechhofer, R. E. A single-sample multiple decision procedure for ranking means of normal populations with known variances. *Annals of Mathematical Statistics 25*

(1954), 16–39.

[27] Bechhofer, R. E., Santner, T. J., and Goldsman, D. M. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons.* John Wiley and Sons, New York, 1995.

[28] Blum, J. R. Multidimensional stochastic approximation method. *Annals of Mathematical Statistics 25* (1954), 737–744.

[29] Boesel, J., Nelson, B. L., and Kim, S. H. Using ranking and selection to 'clean up' after simulation optimization. *Operations Research 51* (2003), (to appear).

[30] Booker, A., Dennis, Jr., J. E., Frank, P., Serafini, D., and Torczon, V. Optimization using surrogate objectives on a helicopter test example. In *Optimal Design* (Philadelphia, 1998), J. Burns and E. Cliff, Eds., SIAM.

[31] Booker, A. J., Dennis, Jr., J. E., Frank, P. D., Moore, D. W., and Serafini, D. B. Managing surrogate objectives to optimize a helicopter rotor design — further experiments. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization* (September 1998). AIAA paper 98-4717.

[32] Booker, A. J., Dennis, Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization 17*, 1 (1999), 1–13.

[33] Box, G. E. P. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics 6* (1957), 81–101.

[34] Carson, Y., and Maria, A. Simulation optimization: Methods and applications. In *Proceedings of the 1997 Winter Simulation Conference* (1997), S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, Eds., pp. 118–126.

[35] Clarke, F. H. *Optimization and Nonsmooth Analysis.* SIAM Classics in Applied Mathematics 5. SIAM, Philadelphia, 1990.

[36] Conn, A. R., Gould, N. I. M., and Toint, P. L. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis 28*, 2 (1991), 545–572.

[37] Davis, C. Theory of positive linear dependence. *American Journal of Mathematics 76*, 4 (1954), 733–746.

[38] Dennis, Jr., J. E., and Torczon, V. Direct search methods on parallel machines. *SIAM Journal on Optimization 1*, 4 (1991), 448–474.

[39] Dennis, Jr., J. E., and Torczon, V. Managing approximation models in optimization. In *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* (1996). Work in Progress paper.

[40] Dolan, E. D., Lewis, R. M., and Torczon, V. On the local convergence of pattern search. *SIAM Journal on Optimization 14*, 2 (2003), 567–583.

[41] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*, 2nd ed. John Wiley & Sons, New York, 2001.

[42] Dudewicz, E. J., and Dalal, S. R. Allocation of observations in ranking and selection. *The Indian Journal of Statistics 37B*, 1 (1975), 28–78.

[43] Epanechnikov, V. Nonparametric estimates of a multivariate probability density. *Theory of Probability and its Applications 14* (1969), 153–158.

[44] Ermoliev, Y. On the method of generalized stochastic gradients and quasi-fejer sequences. *Cybernetics 5* (1969), 208–220.

[45] Fletcher, R., and Leyffer, S. Nonlinear programming without a penalty function. *Mathematical Programming 91*, 2 (2002), 239–269.

[46] Fu, M. C. Optimization via simulation: A review. *Annals of Operations Research 53* (1994), 199–247.

[47] Fu, M. C. Simulation optimization. In *Encyclopedia of Operations Research and Management Science* (Boston, MA, 2001), S. I. Gass and C. M. Harris, Eds., Kluwer Academic, pp. 756–759.

[48] Fu, M. C. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing 14*, 3 (2002), 192–215.

[49] Gelfand, S. B., and Mitter, S. K. Simulated annealing with noisy or imprecise energy measurements. *Journal of Optimization Theory and Applications 62* (1989), 49–62.

[50] Gerencsér, L., Hill, S. D., and Vágó, Z. Optimization over discrete sets via SPSA. In *Proceedings of the 38th Conference on Decision & Control* (1999), IEEE, pp. 1791–1795.

[51] Gerencsér, L., Hill, S. D., and Vágó, Z. Optimization over discrete sets via SPSA. In *Proceedings of the 1999 Winter Simulation Conference* (Piscataway, New Jersey, 1999), P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, Eds., Institute of Electrical and Electronics Engineers, pp. 466–470.

[52] Glynn, P. W. Likelihood ratio gradient estimation: An overview. In *Proceedings of the 1987 Winter Simulation Conference* (Piscataway, New Jersey, 1987), A. Thesen, H. Grant, and W. D. Kelton, Eds., Institute of Electrical and Electronics Engineers, pp. 366–375.

[53] Goldsman, D., and Nelson, B. L. Comparing systems via simulation (chap. 8). In *Handbook of Simulation* (New York, 1998), J. Banks, Ed., John Wiley and Sons, pp. 273–306.

[54] Hajela, P. Nongradient methods in multidisciplinary design optimization–status and potential. *Journal of Aircraft 36*, 1 (1999), 255–265.

[55] Härdle, W. *Applied Nonparametric Regression.* Cambridge University Press, New York, 1990.

[56] Hedlund, H. E., and Mollaghasemi, M. A genetic algorithm and an indifference-zone ranking and selection framework for simulation optimization. In *Proceedings of the 2001 Winter Simulation Conference* (Piscataway, New Jersey, 2001), B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, Eds., Institute of Electrical and Electronics Engineers, pp. 417–421.

[57] Ho, Y. C. A survey of the perturbation analysis of discrete event dynamic systems. *Annals of Operations Research 3* (1985), 393–402.

[58] Hock, W., and Schittkowski, K. *Test Examples for Nonlinear Programming Codes.* Springer-Verlag, Berlin, Heidelberg, New York, 1981. Lecture Notes in Economics and Mathematical Systems No. 187.

[59] Hong, L. J., and Nelson, B. L. An indifference-zone selection procedure with minimum switching and sequential sampling. In *Proceedings of the 2003 Winter Simulation Conference* (Piscataway, New Jersey, 2003), S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, Eds., Institute of Electrical and Electronics Engineers, pp. 474–480.

[60] Hooke, R., and Jeeves, T. A. "Direct search" solution of numerical and statistical problems. *Journal of the Association of Computing Machinery 8* (1961), 212–229.

[61] Humphrey, D. G., and Wilson, J. R. A revised simplex search procedure for stochastic simulation response-surface optimization. In *Proceedings of the 1998 Winter Simulation Conference* (Piscataway, New Jersey, 1998), D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, Eds., Institute of Electrical and Electronics Engineers, pp. 751–759.

[62] Humphrey, D. G., and Wilson, J. R. A revised simplex search procedure for stochastic simulation response surface optimization. *INFORMS Journal on Computing 12*, 4 (2000), 272–283.

[63] Jacobson, S. H., and Schruben, L. W. Techniques for simulation response optimization. *Operations Research Letters 8* (1989), 1–9.

[64] Jin, R., Chen, W., and Simpson, T. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Design Optimization 23* (2001), 1–13.

[65] Jin, R., Du, X., and Chen, W. The use of metamodeling techniques for optimization under uncertainty. In *Proceedings of Design Engineering Technical Conferences* (Pittsburgh, Pennsylvania, September 2001).

[66] Joshi, S., Sherali, H. D., and Tew, J. D. An enhanced response surface methodology (RSM) algorithm using gradient deflection and second-order search strategies. *Computers and Operations Research 25*, 7/8 (1998), 531–541.

[67] Kelahan, R. C., and Gaddy, J. L. Application of the adaptive random search to discrete and mixed integer optimization. *International Journal for Numerical Methods in Engineering 12* (1978), 289–298.

[68] Kiefer, J., and Wolfowitz, J. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics 23* (1952), 462–466.

[69] Kim, S. H., and Nelson, B. L. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation 11*, 3 (2001), 251–273.

[70] Kleywegt, A. J., and Shapiro, A. Stochastic optimization (chap. 101). In *Handbook of Industrial Engineering,* 3rd Edition (New York, 2001), G. Salvedy, Ed., John Wiley, pp. 2625–2650.

[71] Klotz, J. H. *A Computational Approach to Statistics.* University of Wisconsin at Madison, 2004. Copyright by Jerome H. Klotz, downloaded from <http://www.stat.wisc.edu/ klotz/> [accessed 4 August 2004].

[72] Koch, P. N., Simpson, T. W., Allen, J. K., and Mistree, F. Statistical approximations for multidisciplinary design optimization: The problem of size. *Journal of Aircraft 36*, 1 (1999), 275–286.

[73] Koch, P. N., Wujek, B., Golovidov, O., and Simpson, T. Facilitating probabilistic multidisciplinary design optimization using kriging approximation models. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis & Optimization* (September 2002). AIAA paper 2002-5415.

[74] Kokkolaras, M., Audet, C., and Dennis, Jr., J. E. Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optimization and Engineering 2*, 1 (2001), 5–29.

[75] Kushner, H. J., and Clark, D. S. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, New York, 1978.

[76] Kushner, H. J., and Yang, J. Stochastic approximation with averaging of the iterates: Optimal asymptotic rate of convergence for general processes. *SIAM Journal on Control and Optimization 31*, 4 (1993), 1045–1062.

[77] Lacksonen, T. Empirical comparison of search algorithms for discrete event simulation. *Computers and Industrial Engineering 40* (2001), 133–148.

[78] Laguna, M., and Marti, R. Neural network prediction in a system for optimizing simulations. *IIE Transactions 34* (2002), 273–282.

[79] Lewis, R. M., and Torczon, V. Rank ordering and positive bases in pattern

search algorithms. Tech. Rep. ICASE 96-71, NASA Langley Research Center, 1996.

[80] Lewis, R. M., and Torczon, V. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization 9*, 4 (1999), 1082–1099.

[81] Lewis, R. M., and Torczon, V. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization 10*, 3 (2000), 917–941.

[82] Lewis, R. M., and Torczon, V. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization 12*, 4 (2002), 1075–1089.

[83] Ljung, L., Pflug, G., and Walk, H. *Stochastic Approximation and Optimization of Random Systems.* Birkhäuser Verlag, Berlin, 1992.

[84] Lucidi, S., Piccialli, V., and Sciandrone, M. An algorithm model for mixed variable programming. Tech. Rep. 17-02, Department of Computer and Systems Science "Antonio Ruberti", University of Rome, 2002.

[85] Lucidi, S., and Sciandrone, M. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization 13*, 1 (2002), 97–116.

[86] Marsden, A. L., Wang, M., Dennis, Jr., J. E., and Moin, P. Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering 5*, 2 (2004), 235–262.

[87] Matyas, J. Random optimization. *Automation and Remote Control 26*, 2 (1965), 246–253.

[88] McKay, M. D., Beckman, R. J., and Conover, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2 (1979), 239–245.

[89] Mebarki, N., and Castagna, P. An approach based on hotelling's test for multicriteria stochastic simulation-optimization. *Simulation Practice and Theory 8* (2000), 341–355.

[90] Mebarki, N., Dussauchoy, A., and Pierreval, H. On the comparison of solutions in stochastic simulation-optimization problems with several performance measures. *International Transactions of Operational Research 5*, 2 (1998), 137–145.

[91] Meckesheimer, M., Booker, A. J., Barton, R. R., and Simpson, T. W. Computationally inexpensive metamodel assessment strategies. *AIAA Journal 40*, 10 (2002), 2053–2060.

[92] Meketon, M. Optimization in simulation: A survey of recent results. In *Proceedings of the 1987 Winter Simulation Conference* (Piscataway, New Jersey, 1987), A. Thesen, H. Grant, and W. Kelton, Eds., Institute of Electrical and Electronics Engineers, pp. 58–67.

[93] Montgomery, D. C. *Design and Analysis of Experiments*, 5th ed. John Wiley & Sons, New York, 2001.

[94] Myers, R. H., and Montgomery, D. C. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed. John Wiley and Sons, New York, 2002.

[95] Nadaraya, E. A. On estimating regression. *Theory of Probability and Its Applications 9* (1964), 141–142.

[96] Nandkeolyar, U., and Christy, D. P. Using computer simulation to optimize flexible manufacturing system design. In *Proceedings of the 1989 Winter Simulation Conference* (Piscataway, New Jersey, 1989), E. A. MacNair, K. J. Musselman, and P. Heidelberger, Eds., Institute of Electrical and Electronics Engineers, pp. 396–405.

[97] Nazin, A. V., Polyak, B. T., and Tsybakov, A. B. Optimal and robust kernel algorithms for passive stochastic approximation. *IEEE Transactions on Information Theory 38*, 5 (1992), 1577–1583.

[98] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The Computer Journal 7*, 4 (1965), 308–313.

[99] Nelson, B. L., Swann, J., Goldsman, D., and Song, W. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research 49*, 6 (2001), 950–963.

[100] Nocedal, J., and Wright, S. J. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[101] Nozari, A., and Morris, J. S. Application of an optimization procedure to steady-state simulation. In *Proceedings of the 1984 Winter Simulation Conference* (Piscataway, New Jersey, 1984), A. Sheppard, U. Pooch, and D. Pegden, Eds., Institute of Electrical and Electronics Engineers, pp. 217–219.

[102] Ólafsson, S. Iterative ranking-and-selection for large-scale optimization. In *Proceedings of the 1999 Winter Simulation Conference* (1999), P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, Eds., pp. 479–485.

[103] Ólafsson, S., and Kim, J. Simulation optimization. In *Proceedings of the 2002 Winter Simulation Conference* (Piscataway, New Jersey, 2002), E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, Eds., Institute of Electrical and Electronics Engineers, pp. 79–84.

[104] Ouali, M. S., Aoudjit, H., and Audet, C. Optimisation des stratégies de maintenance intégration á la production. *Journal Européen des Systèmes Automatisés 37*, 5 (2003), 587–605.

[105] Parzen, E. On estimation of a probability density and mode. *Annals of Mathematical Statistics 35* (1962), 1065–1076.

[106] Pegden, C. D., and Gately, M. P. Decision optimization for GASP IV simulation models. In *Proceedings of the 1977 Winter Simulation Conference* (Piscataway, New Jersey, 1977), Institute of Electrical and Electronics Engineers, pp. 127–133.

[107] Pegden, C. D., and Gately, M. P. A decision-optimization module for SLAM. *Simulation 34*, 1 (1980), 18–25.

[108] Pichitlamken, J. *A Combined Procedure for Optimization Via Simulation.* PhD thesis, Northwestern University, 2002.

[109] Pichitlamken, J., and Nelson, B. L. Selection-of-the-best procedures for optimization via simulation. In *Proceedings of the 2001 Winter Simulation Conference* (Piscataway, New Jersey, 2001), B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, Eds., Institute of Electrical and Electronics Engineers, pp. 401–407.

[110] Pichitlamken, J., and Nelson, B. L. A combined procedure for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation 13*, 2 (2003), 155–179.

[111] Polyak, B. T., and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization 30*, 4 (1992), 838–855.

[112] Reeves, C. R., and Beasley, J. E. Introduction (chapter 1). In *Modern Heuristic Techniques for Combinatorial Problems* (New York, 1993), C. R. Reeves, Ed., John Wiley and Sons, pp. 1–19.

[113] Resnick, S. I. *A Probability Path.* Birkhäuser, Boston, 1998.

[114] Rinott, Y. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics A7*, 8 (1978), 799–811.

[115] Robbins, H., and Monro, S. A stochastic approximation method. *Annals of Mathematical Statistics 22* (1951), 400–407.

[116] Royston, J. P. An extension of shapiro and wilk's w test for normality to large samples. *Applied Statistics 31* (1982), 115–124.

[117] Ruszczyński, A., and Shapiro, A. Stochastic programming models. In *Stochastic Programming (Handbooks in Operations Research and Management Science)* (Amsterdam, 2003), A. Ruszczyński and A. Shapiro, Eds., Elsevier Science, pp. 1–64.

[118] Sadegh, P. Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation. *Automatica 33*, 5 (1997), 889–892.

[119] Safizadeh, M. H. Optimization in simulation: Current issues and the future outlook. *Naval Research Logistics 37* (1990), 807–825.

[120] Salcedo, R. L. Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search. *Industrial & Engineering Chemistry Research 31*, 1 (1992), 262–273.

[121] SAS Institute, Inc. *JMP Statistics and Graphics Guide, Version 5.1.* Cary, N.C., 2003.

[122] Schittkowski, K. *More Test Examples for Nonlinear Programming Codes.* Springer-Verlag, Berlin, Heidelberg, New York, 1987. Lecture Notes in Economics and Mathematical Systems No. 282.

[123] Schmidt, J. W., and Taylor, R. E. *Simulation and Analysis of Industrial Systems.* Irwin, Homewood, IL, 1970.

[124] Schruben, L. W. Simulation sensitivity analysis: A frequency domain approach. In *Proceedings of the 1981 Winter Simulation Conference* (Piscataway, New Jersey, 1981), T. I. Oren, C. M. Delfosse, and C. M. Shub, Eds., Institute of Electrical and Electronics Engineers, pp. 455–459.

[125] Shapiro, S. S., and Wilk, M. B. An analysis of variance test for normality. *Biometrika 52* (1965), 591–611.

[126] Sheskin, D. J. *Handbook of Parametric and Nonparametric Statistical Procedures*, 2nd ed. Chapman & Hall / CRC, Boca Raton, FL, 2000.

[127] Siefert, C. M., Torczon, V., and Trosset, M. W. Model-assisted pattern search methods for optimizing expensive computer simulations. In *Proceedings of the Section on Physical and Engineering Sciences, American Statistical Association* (2002).

[128] Simpson, T. W. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal 39*, 12 (December 2001), 2233–2241.

[129] Simpson, T. W., Mauery, T. M., Korte, J. J., and Mistree, F. Comparison of response surface and kriging models for multidisciplinary design optimization. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization* (September 1998). AIAA paper 98-4755.

[130] Smith, D. E. An empirical investigation of optimum-seeking in the computer simulation situation. *Operations Research 21* (1973), 475–497.

[131] Solis, F. J., and Wets, R. J.-B. Minimization by random search techniques. *Mathematics of Operations Research 6*, 1 (1981), 19–30.

[132] Spall, J. C. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *Proceedings of the 1987 American Control Conference* (Minneapolis, MN, 1987), pp. 1161–1167.

[133] Spall, J. C. A stochastic approximation algorithm for large-dimensional systems

in the kiefer-wolfowitz setting. In *Proceedings of the IEEE Conference on Decision and Control* (1988), pp. 1544–1548.

[134] Spall, J. C. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* John Wiley and Sons, Hoboken, New Jersey, 2003. MATLAB code available online via <http://www.jhuapl.edu/ISSO/> [accessed March 19, 2004].

[135] Specht, D. F. A general regression neural network. *IEEE Transactions on Neural Networks 2*, 6 (November 1991), 568–576.

[136] Spendley, W., Hext, G. R., and Himsworth, F. R. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics 4*, 4 (1962), 441–461.

[137] Swisher, J. R., Hyden, P. D., Jacobson, S. H., and Schruben, L. W. A survey of simulation optimization techniques and procedures. In *Proceedings of the 2000 Winter Simulation Conference* (Piscataway, New Jersey, 2000), J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, Eds., Institute of Electrical and Electronics Engineers, pp. 119–128.

[138] Swisher, J. R., Hyden, P. D., Jacobson, S. H., and Schruben, L. W. A survey of recent advances in discrete input parameter discrete-event simulation optimization. *IIE Transactions 36* (2004), 591–600.

[139] Swisher, J. R., and Jacobson, S. H. A survey of ranking, selection, and multiple comparison procedures for discrete-event simulation. In *Proceedings of the 1999 Winter Simulation Conference* (Piscataway, New Jersey, 1999), P. Farrington, H. Nembhard, D. Sturrock, and G. Evans, Eds., Institute of Electrical and Electronics Engineers, pp. 492–501.

[140] Swisher, J. R., Jacobson, S. H., and Yücesan, E. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation 13*, 2 (2003), 134–154.

[141] Tomick, J. J. *On Convergence of the Nelder-Mead Simplex Algorithm for Unconstrained Stochastic Optimization.* PhD thesis, The Pennsylvania State University, Department of Statistics, May 1995.

[142] Tomick, J. J., Arnold, S. F., and Barton, R. R. Sample size selection for improved nelder-mead performance. In *Proceedings of the 1995 Winter Simulation Conference* (Piscataway, New Jersey, 1995), C. Alexopoulous, K. Kang, W. R. Lilegdon, and D. Goldsman, Eds., Institute of Electrical and Electronics Engineers, pp. 341–345.

[143] Torczon, V. On the convergence of pattern search algorithms. *SIAM Journal on Optimization 7*, 1 (1997), 1–25.

[144] Torczon, V., and Trosset, M. W. From evolutionary operation to parallel

direct search: Pattern search algorithms for numerical optimization. *Computing Science and Statistics 29*, 1 (1997), 396–401.

[145] Torczon, V., and Trosset, M. W. Using approximations to accelerate engineering design optimization. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization* (September 1998). AIAA paper 98-4800.

[146] Trosset, M. W. On the use of direct search methods for stochastic optimization. Tech. Rep. TR00-20, Department of Computational and Applied Mathematics, Rice University, Houston Texas, June 2000.

[147] Trosset, M. W., and Torczon, V. Numerical optimization using computer experiments. Tech. Rep. ICASE 97-38, NASA Langley Research Center, 1997.

[148] Wang, I.-J., and Spall, J. C. A constrained simultaneous perturbation stochastic approximation algorithm based on penalty functions. In *Proceedings of the American Control Conference* (1999), pp. 393–399.

[149] Wardi, Y. Stochastic algorithms with armijo stepsizes for minimization functions. *Journal of Optimization Theory and Applications 64*, 2 (1990), 399–417.

[150] Wasserman, P. D. *Neural Computing.* Van Nostrand Reinhold, New York, 1989.

[151] Watson, G. S. Smooth regression analysis. *Sankhyā, Series A 26* (1964), 359–372.

[152] Wright, M. H. What, if anything, is new in optimization? Tech. Rep. 00-4-08, Lucent Technologies, Bell Laboratories, Murray Hill, New Jersey, June 2000.

[153] Yakowitz, S. J., and Fisher, L. On sequential search for the maximum of an unknown function. *Journal of Mathematical Analysis and Applications 41* (1973), 234–259.

[154] Yan, D., and Mukai, H. Stochastic discrete optimization. *SIAM Journal on Control and Optimization 30*, 3 (1992), 594–612.

[155] Zhigljavsky, A. A. *Theory of Global Random Search.* Kluwer Academic, Boston, 1991.

Vita

Major Todd A. Sriver was born in Plymouth, Indiana and raised in South Bend, Indiana. He graduated from South Bend's Riley High School in 1986 and then attended Purdue University in West Lafayette, Indiana. He graduated with a Bachelor of Science degree in Aeronautical and Astronautical Engineering in 1990. He was commissioned in the United States Air Force on 22 September 1993 upon graduation from Officer Training School at Lackland Air Force Base, Texas.

In his first assignment, Major Sriver served as an astronautical engineer and Chief of Requirements Analysis in the Engineering Directorate of Detachment 2, Space and Missiles Systems Center at Onizuka Air Station, California. In August 1996, he was reassigned to the Air Force Institute of Technology (AFIT) at Wright-Patterson Air Force Base, Ohio, to work on a master of science degree in Operations Research, graduating in March of 1998 as a distinguished graduate. From April 1998 to August 2001, Major Sriver served as Chief Scientist for the 33d Flight Test Squadron, Air Mobility Command's sole operational test and evaluation agency located at Fort Dix, New Jersey. In September 2001, he returned to AFIT to work on a doctorate in Operations Research. Upon graduation in September 2004, Major Sriver will be reassigned to the Air Force Personnel Operations Agency at the Pentagon.

Index

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
09-2004	**Doctoral Dissertation**	Sep 2001 – Sep 2004

4. TITLE AND SUBTITLE PATTERN SEARCH RANKING AND SELECTION ALGORITHMS FOR MIXED-VARIABLE OPTIMIZATION OF STOCHASTIC SYSTEMS	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Sriver, Todd A., Major, USAF	**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Street, Building 642 WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DS/ENS/04-02
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Juan R. Vasquez, Major, USAF, Ph.D. Air Force Office of Scientific Research 4015 Wilson Blvd, Room 173 Arlington, VA 22203-1954	**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A new class of algorithms is introduced and analyzed for bound and linearly constrained optimization problems with stochastic objective functions and a mixture of design variable types. The generalized pattern search (GPS) class of algorithms is extended to a new problem setting in which objective function evaluations require sampling from a model of a stochastic system. The approach combines GPS with ranking and selection (R&S) statistical procedures to select new iterates. The derivative-free algorithms require only black-box simulation responses and are applicable over domains with mixed variables (continuous, discrete numeric, and discrete categorical) to include bound and linear constraints on the continuous variables. A convergence analysis for the general class of algorithms establishes almost sure convergence of an iteration subsequence to stationary points appropriately defined in the mixed-variable domain. Additionally, specific algorithm instances are implemented that provide computational enhancements to the basic algorithm. Implementation alternatives include the use of modern R&S procedures designed to provide efficient sampling strategies and the use of surrogate functions that augment the search by approximating the unknown objective function with nonparametric response surfaces. In a computational evaluation, six variants of the algorithm are tested along with four competing methods on 26 standardized test problems. The numerical results validate the use of advanced implementations as a means to improve algorithm performance.

15. SUBJECT TERMS

Pattern Search, Ranking and Selection, Stochastic Optimization, Mixed Variable Programming, Simulation Optimization, Kernel Regression, Surrogate Functions

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			James W. Chrissis, AFIT/ENS
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	251	(937) 255-3636, ext 4606; e-mail: James.Chrissis@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18